NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A FRAMEWORK FOR COLLABORATIVE QUADROTOR – GROUND ROBOT MISSIONS

by

Georgios Milionis

December 2011

Thesis Advisor: Oleg Yakimenko
Thesis Co-Advisor: Richard Harkins

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2011	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE A Framework for Collaborative Quadrotor – Ground Robot Missions	**5. FUNDING NUMBERS**
6. AUTHOR(S) Georgios Milionis	

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number _____N/A_____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The thesis proposes a real-time control algorithm for the cooperation of a joint team consisting of a Quadrotor and a Ground robot for coordinated ISR missions. The intended application focuses on indoor environments, where Global Positioning System signals are unreliable or simply unavailable so that the control algorithms must rely on local sensor information. The thesis describes the appropriate set up of the lab and includes simulations using a full dynamic model of the quadrotor and robot, demonstrating the suitability of the implemented and the proposed control scheme into a waypoint navigation scenario.

The implemented controller uses the Linear Quadratic Regulator method imposed into five different channels; pitch, roll, yaw, x-y position and height, configured to the appropriate gains for smoother following of the trajectory. The proposed control scheme incorporates three key aspects of autonomy; trajectory planning, trajectory following and collaboration of the two vehicles. Using the differentially-flat dynamics property of the system, the trajectory optimization is posed as a non-linear constrained optimization within the output space in the virtual domain, not explicitly related to the time domain. A suitable parameterization using a virtual argument as opposed to time is applied, which ensures initial and terminal constraint satisfaction. The speed profile is optimized independently, followed by the mapping to the time domain achieved using a speed factor.

14. SUBJECT TERMS Quadrotor, Ground Robot, Cooperation, Localization System, Quanser, Optitrack cameras, linearization, LQR, Waypoint Navigation, Direct method, Trajectory Generator, Trajectory Following, Optimization, Inverse Dynamics.	**15. NUMBER OF PAGES** 173
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

**A FRAMEWORK FOR COLLABORATIVE
QUADROTOR – GROUND ROBOT MISSIONS**

Georgios Milionis
Lieutenant, Hellenic Navy
B.S., Greek Naval Academy, 2002

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN
APPLIED PHYSICS AND MECHANICAL ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL
December 2011**

Author: Georgios Milionis

Approved by: Oleg Yakimenko
               Thesis Advisor

               Richard Harkins
               Thesis Co-Advisor

               Knox Millsaps
               Chair, Department of Mechanical Engineering

               Andres Larraza
               Chair, Department of Physics

iii

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The thesis proposes a real-time control algorithm for the cooperation of a joint team consisting of a Quadrotor and a Ground robot for coordinated ISR missions. The intended application focuses on indoor environments, where Global Positioning System signals are unreliable or simply unavailable so that the control algorithms must rely on local sensor information. The thesis describes the appropriate set up of the lab and includes simulations using a full dynamic model of the quadrotor and robot, demonstrating the suitability of the implemented and the proposed control scheme into a waypoint navigation scenario.

The implemented controller uses the Linear Quadratic Regulator method imposed into five different channels; pitch, roll, yaw, x-y position and height, configured to the appropriate gains for smoother following of the trajectory. The proposed control scheme incorporates three key aspects of autonomy; trajectory planning, trajectory following and collaboration of the two vehicles. Using the differentially-flat dynamics property of the system, the trajectory optimization is posed as a non-linear constrained optimization within the output space in the virtual domain, not explicitly related to the time domain. A suitable parameterization using a virtual argument as opposed to time is applied, which ensures initial and terminal constraint satisfaction. The speed profile is optimized independently, followed by the mapping to the time domain achieved using a speed factor.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

xi

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ISR	Intelligence, Surveillance and Reconnaissance
MIO	Maritime Interdiction Operations
UAV	Unmanned Air Vehicles
UGV	Unmanned Ground Vehicles
UMV	Unmanned Maritime Vehicles
USV	Unmanned Surface Vehicles
UUV	Unmanned Underwater Vehicles
UV	Unmanned Vehicle
LQR	Linear Quadratic Regulator
QuarC	Quanser Real-time Control
DAC	Data Acquisition Card
PID	Proportional and Integral Derivative
IMU	Inertial Measurement Unit
ESC	Electronic Speed Controller
GPS	Global Positioning System
QCM	Quanser Controller Module
HIL	Hardware In the Loop
IR/EO	InfraRed/ Electoroptical
PWM	Pulse Width Modulation
ICC	Instantaneous Center of Curvature
IDVD	Inverse Dynamics in Virtual Domain
FL	Feedback Linearization
USB	Universal Serial Bus
PCB	Printed Circuit Board
DOF	Degrees Of Freedom

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my advisors Dr. Oleg Yakimenko and S.L. Richard Harkins for their continuous help, support and meaningful advice throughout the whole thesis research.

THIS PAGE INTENTIONALLY LEFT BLANK

I.   INTRODUCTION

A.   GENERAL

An Unmanned Vehicle (UV) is a power driven vehicle that can be autonomously or remotely operated and controlled. Normally, it can be deployed or recovered repeatedly with various types of payload. Unmanned Vehicles are often divided in three major categories, (1) Air (Unmanned Air Vehicles-UAVs), (2) Ground (Unmanned Ground Vehicles-UGVs) and (3) Maritime (Unmanned Maritime Vehicles- UMVs). The Maritime category can be divided into Surface (Unmanned Surface Vehicles-USVs) and Underwater (Unmanned Underwater Vehicles-UUV) functional areas. There are more divisions depending on whether the UVs are military or commercial oriented. If military, the branch and missions dictate the functional role, of which, Intelligence, Surveillance and Reconnaissance (ISR) missions are the most common. Many UVs are also used as force-multipliers. First wave strikes, aerial refueling, Maritime Interdiction Operations (MIO), autonomous payload recovery, search and rescue operations and medical purposes come to mind.

Autonomous cooperation of multiple unmanned vehicles is a major concern for several of those functional roles. Various advantages are observed concerning different types of vehicles and missions. For example, two or more UAVs cooperating together in order to identify a moving target on the ground; UAVs and ground robots conducting ISR, with the UAV as a leader and the robot assisting as a follower or inversely.

The dynamic modeling and control of these vehicles is the focus of this thesis. Rigorous analysis of the dynamics and kinematics has been done in order to set up the models for designing the controllers. The model of the ground robot is extracted from the forward and inverse kinematics of the vehicle. Using multiple sensors, there was an effort to apply several techniques for the control of the UVs. The optimization methods include the Linear Quadratic Regulator (LQR) and the use of differential flatness property of the vehicle's dynamics, for the trajectory planning and optimization in the output space.

In real applications, like the dynamic environment of use ISR missions, quasi-optimal trajectories proved to be more effective, by achieving full autonomous control with trajectory planning and path following.

B.   MISSIONS

There are many recent developments that concern unmanned systems. Most of them are used for dangerous and "dirty" human jobs. Facts reveal that a wide range of requirements and capabilities have been developed and fulfilled toward various warfare specifications, material requirements and interoperability.

Since UVs can continuously meet the operator's requirements and needs during or before the battle, it is expected in the near future that the unmanned systems will be an indispensable support element in a wide area of missions. Most of those listed for any category of UVs is in the Table 1. It is quite noticeable that some types of mission can be conducted by a certain type of unmanned vehicle and some others by all of them, within its capabilities.

It is often more effective and desirable for a combination of multiple vehicles to be used in a mission. However, there is a challenge. How can this cooperation of two or more vehicles could be achieved successfully in real-time operations? Also, the question of which vehicles should be chosen for a specific operation, especially for the UAVs, since there are a lot of candidates? Observing the characteristics of the quadrotor helicopter, it seems right to compare it with some of the already developed unmanned air vehicles in order to prove how useful it can be toward the ISR missions and show the advantages of our choice.

UAV	UGV	UUV/USV
Reconnaissance	Reconnaissance	ISR
Precision Target Location and Designation	Precision Target Location and Designation	Communication / Navigation Network Node
Signals Intelligence	CBRNE Reconnaissance	Inspection / Identification
Special Operation Forces (SOF) Team Resupply	Mine detection / Countermeasures	Open Ocean Anti-Submarine Warfare (ASW)
Battle Management	Weaponization / Strike	Payload Delivery
Communication / Data Relay	Battle Management	CBRNE Reconnaissance
CBRNE Reconnaissance	Communication / Data Relay	SOF Resupply
Combat Search and Rescue	Signals Intelligence	Strike
Weaponization / Strike	Covert Sensor Insertion	Littoral Surface Warfare
Electronic Warfare	Littoral Warfare	Mine Countermeasures (MCM)
Mine detection / Countermeasures	Counter CCD Camera	Information Operations
Information Warfare		Time Critical Strike
Digital Mapping		Digital Mapping
Covert Sensor Insertion		Oceanography
Decoy / Pathfinder		Decoy Pathfinder
GPS Pseudolite		Bottom Topography
Littoral Undersea Warfare		

Table 1.   Mission for Unmanned Vehicles

C. QUADROTOR TOWARD OTHER UAVs

a. The limitations of fixed-wing unmanned aircrafts, as far as ISR missions are concerned (the inability to hover or fly at low speeds), have motivated the researchers to look at the use of Rotary-wing unmanned vehicles. Additionally, the ability to hover over an assigned position creates the opportunity to use various sensors, common or experimental ones, attached to the platform. They can be used to collect data, conducting surveillance for ISR missions. Moreover, the ability of the rotary wing UAVs for vertical takeoffs and landings can reduce the launch and recover footprint and eliminate the need of launching mechanisms like catapults and rails that is most common in any fixed-wing UAVs.

b. However, the large mechanical complexity of the helicopter is major concern. A complex hub in the main rotor is needed for lift and for pitch alterations. Also, the vertical tail rotor is used to compensate for the reaction torque on the fuselage, comes from the main rotor. The quadrotor, however, consists of a pair of counter rotating rotors, so a tail rotor, is not only required but it offers better mobility and increases the payload capability. The rotor is also surrounding by a protective frame, so additional safety can be enhanced.

c. It will be useful to compare the different configurations and mechanisms in UAVs that already were used for research purposes. Each of them present advantages and drawbacks as far as maneuverability, compactness, mechanics and aerodynamics complexity, payload capability and survivability is concerned. In Table 2, a small comparison between those different VTOL vehicle concepts is described. It is noticeable that the quadrotor turns out to be the best configuration among them all.

d. However, there are some disadvantages. The quadrotor experiences dynamic instabilities and has higher sensitivity to disturbances. These facts increase difficulty to the control implementation. Also, the extra motors it carries, gives the ability to carry larger payload, but results in the increase of energy consumption.

CONCEPT	VTOL VEHICLE							
	Single Rotor (A.V. de Rostyne)	Axial Rotor (Maryland University)	Coaxial Rotors (ETHZ)	Tandem Rotors (Heudiasyc)	Quad Rotor (ETHZ)	Blimp (EPFL)	Bird Like (Caltech)	Insect Like (UC Berkeley)
Power Cost	2	2	2	2	1	4	3	3
Control Cost	1	1	4	2	3	3	2	1
Payload Volume	2	2	4	3	3	1	2	1
Maneuverability	4	2	2	3	3	1	3	3
Mechanics Simplicity	1	3	3	1	4	4	1	1
Aerodynamics Complexity	1	1	1	1	4	3	1	1
Low Speed Flight	4	3	4	3	4	4	2	2
High Speed Flight	2	4	1	2	3	1	3	3
Miniaturization	2	3	4	2	3	1	2	4
Survivability	1	3	3	1	1	3	2	3
Stationary Flight	4	4	4	4	4	3	1	2
Total	24	28	32	24	**33**	28	22	24

Table 2.   VTOL Concept comparison (1=Bad, 4= Very Good). After [1].

D.   RELATED WORK – RECENT DEVELOPMENTS

Today, a lot of universities are focusing their research projects on quadrotor UAVs. At the same time, many companies have designed commercial quadrotors in a very efficient and effective way. Draganfly Innovations Inc. in Canada is one of the companies, they have produced many great products, like the quadrotor helicopter Draganflyer X4 (Figure 1) and the six-rotor helicopter Draganflyer X6 and Draganflyer RC helicopters.

The Draganflyer X4 is a stable UAV platform [2] equipped with multiple sensors including gyroscopes, magnetometers, accelerometers, barometric pressure sensors and wireless video camera. It is used by many universities, such as the Massachusetts Institute of Technology (MIT) [3], Boeing Research and Technology [4], Vanderbilt University [5], and Concordia University in Canada [6].

Figure 1.   Draganflyer. After [1].

Draganflyer RC helicopters were used in developing an autonomous Control System (VECPAV) shown in Figure 2 [4] by Vanderbilt University. They controlled the helicopter by receiving and processing position and motion data from a sensor and sending various control commands through a radio transmitter. As reported in [7] and [8], Draganflyer X6 helicopter is being widely used for police actions, too. Mesa County

Sheriff is the first Public Safety Agency to receive an FAA Certificate of Authorization (COA) to operate the Draganflyer X6 helicopter for law enforcement use in over a 3,300 square mile area. May Regina's police force, investigating a homicide, used the Draganflyer X6 UAV helicopter to obtain aerial pictures and video of the crime scene.
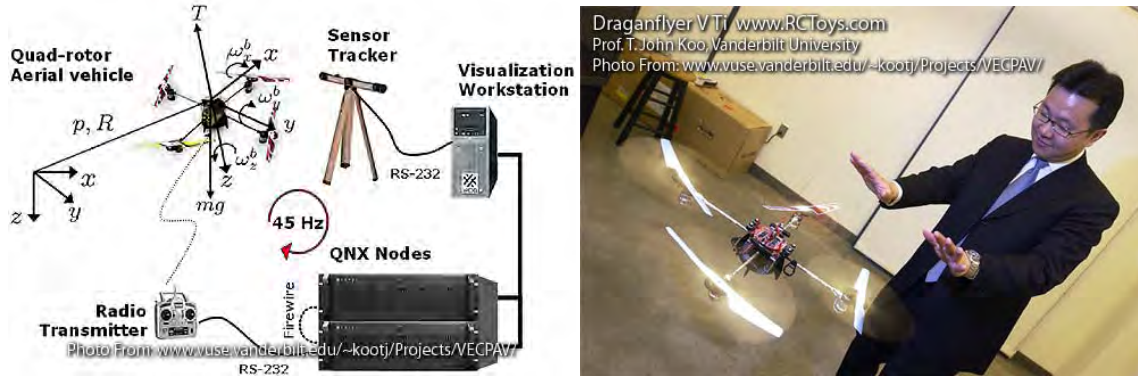
Figure 2.   Vanderbilt University's Embedded Computing Platform. After [3].

Stanford university created a quadrotor helicopter platform (Stanford Testbed of Autonomous Rotorcraft for Multi-Agent Control II - STARMAC II), shown in Figure 3, and applied a PID controller for attitude/altitude control in outside environment where disturbances are unpredicted. The first efforts were not successful since the control of the vehicle flight was not accurate,. Further improvements of the controller are already considered [9].

Figure 3.   STARMAC II quad-rotor UAV. After [8].

In Ecole Polytechnique Federale De Lausanne (EPFL), a quadrotor helicopter called Omni-directional Stationary Flying Outstretched Robot (OS4) was designed for fully autonomous operation using many different control methods. Using the Lyapunov theory, linear controllers like Proportional and Integral Derivative (PID) and Linear Quadratic Regulator (LQR), backstepping and sliding mode methods, their research proved successful by using Integral backstepping where autonomous hovering with altitude control and autonomous take-off and landing were established [1].
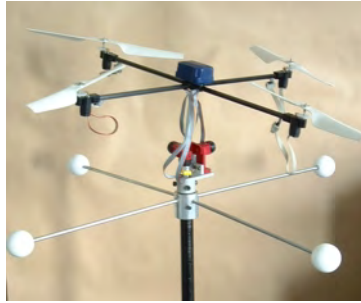
Figure 4.   OS4 Test Bench. After [9].

Moreover, Massachusetts Institute of Technology (MIT) developed a Real-time indoor Autonomous Vehicle test Environment (RAVEN) shown in Figure 5, consisting of ground UVs and UAV and a motion capture system for tracking in order to conduct autonomously multi-vehicle missions. It was a successfully developed and tested system where pilot's assigned tasks can be performed in real-time by a single or multiple vehicles. The vehicles followed waypoints created through a trajectory generation algorithm. RAVEN allows an operator to control up to ten UVs simultaneously [10].
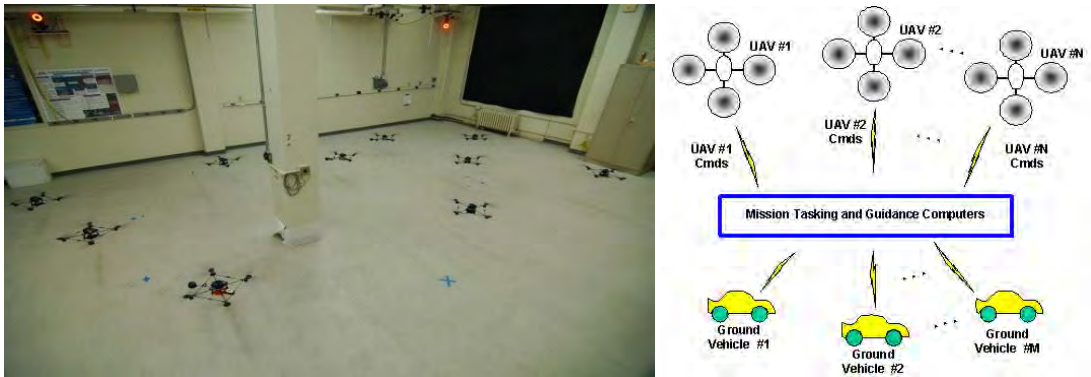
Figure 5.   RAVEN Project. After [11].

From 2007 the Networked Autonomous Vehicles Lab. (NAVL) of Concordia University used different quadrotor UAVs such as Draganflyer X4 and X-Qball from Quanser Innovations INC Company, as well as several wheeled robots as shown in Figure 6. They used various control techniques, focusing mostly on fault-tolerant cooperative control of these vehicles [6].

Figure 6.   Concordia NAVL Project. After [5].

Research has been done by Virginia Tech University [11], only for cooperation of UGVs whereas team of mini ground heterogeneous autonomous vehicles (MUVs) were developed for urban search and rescue in both indoor and outdoor environments by utilizing onboard computers and multiple sensors. Autonomous navigation, search, tracking, localization and mapping (STLAM) as well as obstacle avoidance accomplished through various experiments and participation in competitions as MAGIC2010. Figure 6 depicts the team of those UGVs and the sensors that are used in each vehicle.
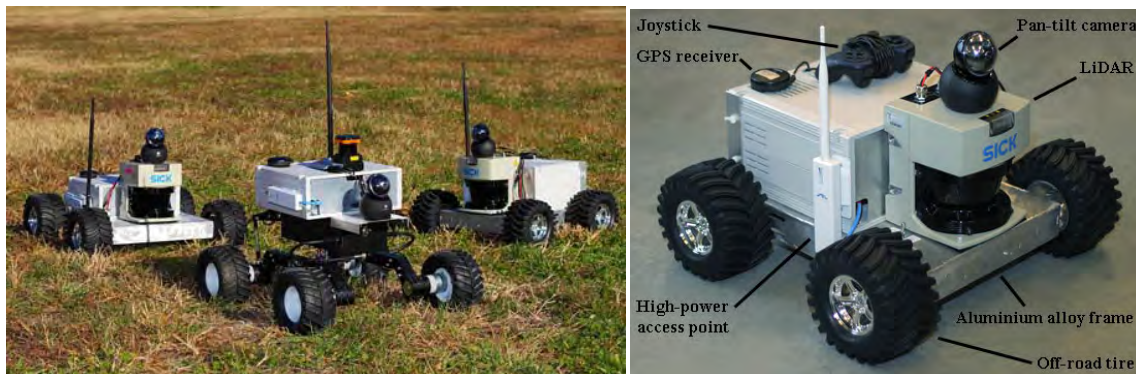
Figure 7.   Virginia Tech University's team of MUVs. After [11].

Various Naval Postgraduate School (NPS) departments have conducted research for autonomous ground vehicles through the years. Small Robot Technology (SMART) program in NPS Physics department AXV LAB many prototype robotic platforms or military applications. One of all the platforms being developed, was called "Bender" and was equipped with a web-cam and several ultrasound sensors so as to move to designated destinations autonomously via waypoint navigation, controlled by a commercial BL2000 microcontroller, which was programmed by Dynamic C language [12]. Another UGV, called "AGB" was a wheeled vehicle, created by MAJ Ben Miller U.S. Army, that utilized an acoustic and IR (infrared) detectors to detect motion, obtaining the capability to report images to remote monitoring stations via a web camera, in order to assist in the interdiction of IED placement [12]. Figure 8 shows these two vehicles.
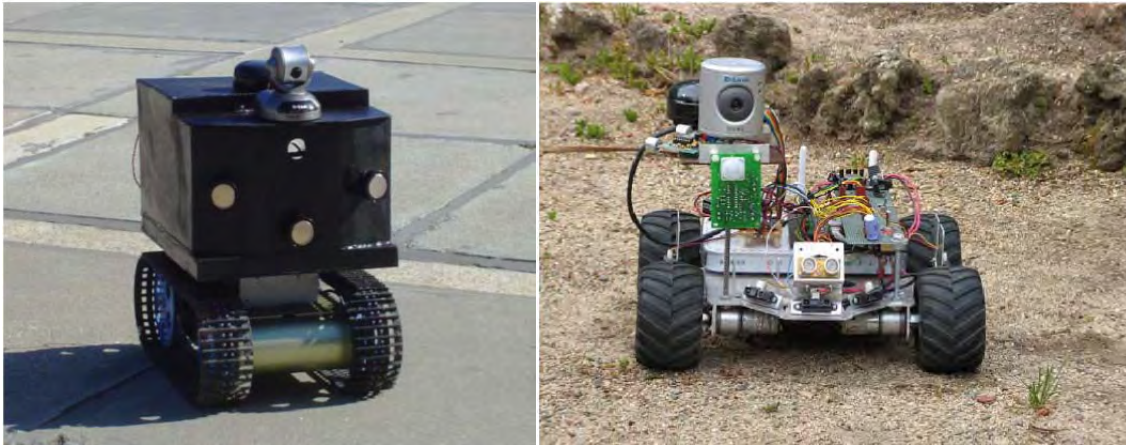
Figure 8.   "Bender" (left) and "AGB" (right) NPS Autonomous Ground Vehicles. After [13].

E. LITERATURE REVIEW

Quadrotors have proved to be quite complex vehicles, but present many advantages for research, including the improvement of their control performance for different flight conditions and through different control techniques and objectives, described in the following reviews [1], [13-16].

One very common technique, used for more than two centuries is the Proportional (P), Integral (I), and Derivative (D) (PID) control method, where the proportional control is used to settle the output signal in direct proportion to the controller input, the integral is used to eliminate any steady state error, and the derivative reduces the overshoot of the system. PID techniques aiming to fault tolerant control of quadrotors have been presented in the following papers [1], [17-20].

Another technique, the feedback linearization (FL) were used by Altug et al. [21] for altitude and Euler angles stabilization when Lee et al. [22] implemented an output feedback (OFB) controller with an observer to a nonlinear system so as to obtain an estimate for the vehicle's velocities. Furthermore, Tayebi et al. [23] used a PD2 (proportional and twice derivative) feedback structure in order to improve the transient performance and remove the disturbances, caused by the feedback linearization of the model. Benallegue et al. [24] showed a way to provide with insensitivity to the uncertainties by combining the FL method with the high-order sliding mode observer which acted also as an estimator of the disturbances.

The stability of the system was always a problem, so Kanellakopoulos et al. introduced the backstepping method [25] so as to hold the nonlinear system [26] for the controller implementation. This technique became very popular through its variations. So, Madani and Benallegue [27] used the backstepping method to control three of the outputs and later combined it with the sliding mode control [28] to solve the chattering phenomenon successfully, while Mian et al. [29] stabilized the altitude simply by adding an integrator.

Several quadrotor vehicles were controlled with Lyapunov theory [30], [31], linear quadratic regulator (LQR) control method [32] and sliding mode control [33].

Castillo et al. [31] presented a controller design based on Lyapunov analysis using a nested saturation algorithm for stabilization of the altitude and the yaw. Real time experiments with autonomous take-off, hovering, and landing. The dynamic model of the quadrotor was obtained with a Lagrange approach while Bouabdallah et al. [1], [30] described the modeling of the OS4 quadrotor with both Langrange and Euler-Newton method and provided results for altitude and attitude control from all of the mentioned techniques, proving that the integral backstepping method was indeed more successful and efficient for his model.

However, this literature mentions control techniques for test benches or quadrotor platforms in a laboratory environment or outside environment with limited disturbances and thus normal flight conditions. There is a possibility that various faults could happen during flight, like actuators and sensors outage (zero output), transient fault and bias failure and control surface defect. This will reduce the safety of the vehicle. Thus nonlinear control schemes should be considered and fault tolerant control (FTC) techniques are to be proposed. Based on FTC methods for aircrafts presented in [34], [35] thirty years ago, Zhang et al. [5] have presented various papers for FTC, with the most comprehensive one, from Zhang and Jiang [36] where they analyzed the background and different control schemes for fault detection and diagnosis (FDD), presenting also the current research activities and future challenges.

Qi et al. [37] used Kalman filter to estimate the states and the parameters of the fault-tolerant control while Zhang and Jiang [38] introduced a two-stage adaptive Kalman filter for the observation of the potential faults and use of the extracted data for reconfiguration of the controller.

Many simulation and experimental work has been done in Concordia University in Canada, using the same quadrotor platform. In his thesis[39], Zhang addressed a FTC design technique based on Lyapunov based nonlinear control techniques in order to produce acceptable performance for potential faults in the quadrotor, while Bilhim, in his own thesis[40] described a gain-scheduled PID controller for fault tolerant control of the quad-rotor.

12

Another concern toward the control of the quadrotor is the ability to conduct various tasks in real time. Yakimenko et al. [41], brought Prof. Taranenko's ideas (direct method of calculus of variations in flight dynamics problems with constraints on states and controls) to a new level and developed algorithms for real-time onboard calculation of quasi-optimal trajectories [42,] [43], [44]. Various tests in combat vehicles and missiles using these algorithms were performed onboard 5th-generation aircraft [45]. Kaminer et al. [46] used this method in 2006 to generate trajectories for landing approach of UAVs and assure flight deconfliction at maneuvering.

Nowadays, more recent researchers are focused on this method, posing the trajectory planning as a nonlinear constrained optimization problem and separately the trajectory (path) following as different problem in order fully autonomy to be achieved. Using the differential flatness property of the equations of motions, this control method uses the optimization in the computation of a new feasible trajectory that meet the dynamic constrains and requirements for the maneuver of the vehicle characterizing by a given performance index. In 2005 Yang et al. [47] addressed a time optimal control of a hovering quadrotor helicopter, where genetic algorithms were adopted. In 2006, Cowling et al. [48], proposed a complete real-time controller for autonomous control of a quadrotor UAV, while Bouktir et al. in 2008 [49] presented a simple direct method for trajectory planning of the quadrotor vehicle. Finally, the same year, Hoffmann et al. [50] in Stanford University, developed an autonomous vehicle trajectory tracking algorithm for the STARMAC platform.

F.   THESIS OBJECTIVES - MOTIVATIONS

Judging from the existed literature, the researchers have succeeded in achieving a quite acceptable performance by applying different control techniques either with linear or nonlinear control theory. The limited experiments of real-time optimal techniques with inverse dynamics for quadrotor vehicles were the motivation to choose this quadrotor UAV for analysis and controller implementation. The importance of the cooperation of two or more vehicles for the successful completion of ISR missions was another urge for this thesis. The thesis begins with the analysis of the dynamic model of the two vehicles, a quadrotor and a ground robot, follows the derivation of the equations of motion in order to set up the state model that will be used for the controllers. Finally, the controller implementation takes place separately for each vehicle. For the quadrotor, it addresses LQR control technique for the four controller models that will be used for the path following of the trajectories that will be generated from a trajectory generator in real time. The trajectory generator will use quasi optimal solution for the generation of the trajectories through parameterization with fifth order polynomials. For the ground robot through forward and inverse kinematics an LQR controller will be used to set up the vehicle as a leader. where after The specific objectives are as follows:

- Derive a mathematical model of the quadrotor UAV, according to its particular physical structure and dynamics;

- Derive a mathematical model of the ground robot, according to forward and inverse kinematics;

- Derive a unified control architecture that enables control of multiple heterogeneous UAV/UGV teams;

- Develop a LQR controller for its vehicle without fault consideration;

- Design a Trajectory Generator for the quadrotor;

- Implement the two models in Matlab – Simulink integrated with Quarc software for the real-time operation of the vehicles , Qball-X4 quadrotor UAV and Qbot ground vehicle

- Finally, perform simulations for the individual control of the vehicles and the cooperation control to analyze the performance of the design technique.

G.   OUTLINE

Having introduced the concept and the objectives of the thesis in the first chapter, the second chapter will address the setup of Quanser control lab, consisting of the experiment platforms UAV QBall X-4 and UGV Qbot and describes their sensors and elements and the whole set up needed for the completion of the experiment.

Chapter III is devoted to modeling two types vehicles. It describes the quadrotor's physical structure and dynamics, and derives its six degree-of-freedom nonlinear mathematical model. The same procedure is described for the ground robot through its forward and inverse kinematics for the derivation of the equation of motion.

Chapter IV addresses the existing and proposed control architecture within the Quanser lab.

The architecture and the simulation process for the cooperation of the two UVs and results of the lab experiments are shown in order to show the effectiveness of the thesis work.

Finally, Chapter VI highlights the conclusion and recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II.   QUANSER LABORATORY SETUP

A.   INTRODUCTION

The laboratory, where the experiments was carried out, is designed in such way, as to give the opportunity to any researcher to perform research in an indoor, safe environment that can easily controlled and / or reconfigured depending on the different demands. It is an open-architecture environment that consists of Quanser's unmanned vehicles, the quadrotor UAV QBall-X4 [51] and the ground robot, Qbot [52], a single ground station and a localization system containing ten cameras, required for tracking the positions of vehicles in the lab space, since no GPS reception is available indoors.

Every component of the lab can be operated from ground station computer, which is equipped with the required research software, including MATLAB / Simulink with real-time control software QuaRC installed and the OptiTrack Tracking Tools program for managing fully integrated with QuaRC, OptiTrack camera system. In addition, wireless communications are achieved through a wireless network adapter, inserted to the ground control station computer while a USB 2.0 port is used for connecting the OptiTrack indoor localization system.

The Quanser, embedded avionics data acquisition card, HiQ on each vehicle, provides high-resolution inertial measurement sensors and motor outputs; while the controllers created from the operator in the host PC can be downloaded and executed on their embedded Gumstix Verdex target computer [53]. A laboratory illustration with a collage of various photographs of the camera set up is shown in Figure 9.
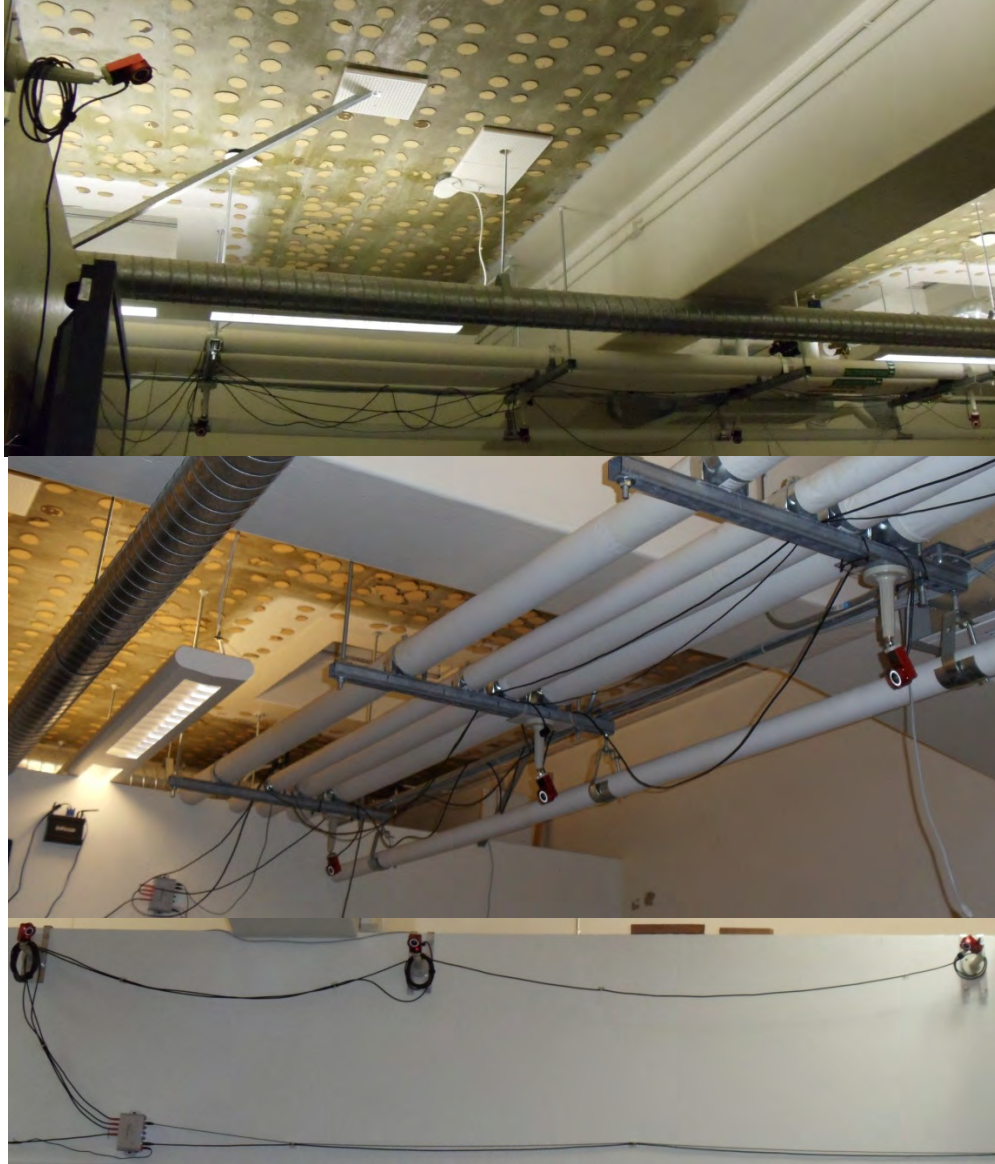
Figure 9.   Implemented Laboratory illustration.

B.   QUANSER REAL-TIME CONTROL (QuaRC) SOFTWARE

Quanser Real-time control (QuaRC) software [52] is a rapid control software for teaching and research applications, created in the paths of its predecessor WinCon, the first real-time software to run Simulink-generated code under Windows. QuaRC is compatible and integrated with Simulink and real-time workshop and gives the opportunity to the operator to design controllers, generate code and execute it avoiding low level programming and pure code writing.

QuaRC runs under Windows XP or Vista and is targeting either the host computer or a remote computer running either Windows or the QNX RTOS. For embedded computers on unmanned vehicles, QuaRC runs under QNX and the Quanser's computer Gumstix Verdex.

So, one or multiple controllers designed in Simulink are able to be converted into real time code through QuaRC and run on different target processors even on different chipsets running different operating systems. QuaRC also provides the operators with the ability to tune the control parameters while the model is running, allowing for rapid design and test iterations, with little to no recompilation required.

The host and target components referred to as QuaRC Host and QuaRC Target in QuaRC software are designed to communicate even via Internet, in order the user to download a controller anywhere and control it from a remote location, tune and configure the control parameters while the controller runs, plot real-time data and save it inside MATLAB.

Low level programming is eliminated by utilizing the QuaRC control software tool, since the number of Simulink blocks to be learned, are reduced. So the controller design becomes the priority avoiding interfacing issues and extreme code-writing.

QuaRC blocks/tools are quite extensible for more systems and commands to be added and even allow a Simulink model to communicate with third party devices, while they provide the mathematical framework for controlling the various devices. The most important ones are:

- Communication blocks for the supported communication protocols, like TCP/IP, UDP, Serial port and Shared memory.

- Hardware-In-the-Loop (HIL) block set, an extensible hardware in-the-loop API used to interface with over 50 data acquisition cards.

- the Vehicle Abstraction Layer (VAL) block set, consisting of a series of blocks that provide a group of high-level primitive commands to the operator, while the VAL deals with the vehicle hardware communication.

- Through the use of the Virtual Reality Toolbox in Simulink and QuaRC, an interactive 3D environment with haptic feedback can be created, allows for simulation or training applications.

C.   GROUND CONTROL STATION

The ground control station is comprised of a PC which is equipped with the required research software, including MATLAB / Simulink with real-time control software QuaRC and the OptiTrack Tracking Tools program the OptiTrack camera system. In addition, wireless communications are achieved through a wireless network adapter, inserted to the ground control station computer while a USB 2.0 port is used for connecting the OptiTrack indoor localization system. The distributed control structure allows the real-time code to be generated from the host controller, wirelessly be sent through QuaRC to any vehicle where it would be compiled and executed on-board it. The ground control station can operate with several vehicle controllers at the same time.

The researcher is able to perform several tasks through the ground control station computer:

- Developing controllers for the navigation of the unmanned vehicles,
- using the OptiTrack Tracking Tools program to calibrate the localization system, in order to provide wireless localization data,
- view Simulink Scopes and several display tools to monitor the vehicles and their status during mission carrying out,
- tune and update the various controller parameters to improve performance during runtime
- log and monitor the mission runtime data.
- conduct vision and image processing

D.   HiQ –GUMSTIX EMBEDDED COMPUTER

The HiQ embedded avionics data acquisition card, shown in Figure 10, developed by Quanser company was integrated with the small form-factor, lightweight Gumstix Verdex embedded computer [53] with wireless communications capabilities that runs a Linux-based operating system in order to achieve measuring the behavior of the unmanned systems being controlled and be able to run generated simulink models designed with QuaRC software.

The Gumstix computer has been configured as a QuaRC target system to create the designed controllers in the ground station computer and download and execute them on the QuaRC target computer without requiring embedded programming. HiQ card provides high-resolution inertial measurement sensors and motor outputs, and is responsible for the reading and writing to vehicle hardware. QuaRC's Hardware-In-the-Loop (HIL) tools give directly access to the Input / Output (I/O) for any supported DAC, including the HiQ.
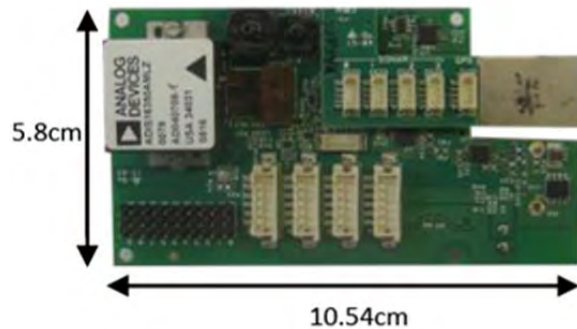
Figure 10.   HiQ embedded avionics data acquisition card. After [55].

The Input / Output of the HiQ card, as it stated in [55] consists of:

- 10 PWM outputs (servo motor outputs)

- 3-axis gyroscope, range configurable for $\pm75^{\circ}$/s, $\pm150^{\circ}$/s, or $\pm300^{\circ}$/s, resolution

- 0.01832°/s/LSB at a range setting of $\pm75^{\circ}$/s

- 3-axis accelerometer, resolution 2.522 mg/LSB

- 10 analog inputs, 12-bit, +3.3V

- 3-axis magnetometer, 0.76923 mGa/LSB

- 4 Maxbotix sonar inputs, 1 inch resolution

- Serial GPS input

- 8 channel RF receiver input

- USB input for on-board camera (up to 9fps)

- 2 pressure sensors, absolute and relative pressure

- Input power 10-20V

E. LOCALIZATION SYSTEM

Indoor laboratories have to utilize alternative localization systems in order to track vehicle positions in the area of the operations, since no access to GPS exists. The OptiTrack Tracking Tools system, used in this laboratory, is created by Natural Point company and designed for indoor applications. It is a camera-based localization and tracking system integrated with QuaRC and consists of a series of at least six motion tracking OptiTrack cameras connected to the ground station computer. The quantity of the cameras depends on the complexity of the environment; with the increase of the quantity to reduce the tracking loss possibility. The QuaRC OptiTrack block set allows operators to track multiple reflective markers simultaneously in 3-D space and/or fixed patterns of markers (rigid bodies) in 6-DOF position and orientation. The features of the OptiTrack vision system as stated in [55] are listed below:

- Up to 16 cameras (Figure 12) can be connected and configured for single or multiple capture volumes, for the experiment ten cameras were chosen.
- Capture volumes up to 400 square feet
- Single point tracking for up to 80 markers, or 10 rigid-body objects
- Typical calibration time is under 5 minutes
- Position accuracy on the order of mm under typical conditions
- USB 2.0 connectivity to ground station PC
- Up to 100 fps tracking

Figure 11.   Natural Point OptiTrack camera, adapted from [55].

More specifically, the ten cameras were chosen to be in the position shown in Figure 9 in order to eliminate the blind areas for every possible scenario that it will be taken place. The connection should have implemented through the right configuration since the data from the cameras can be transferred successfully if the cables are up to 5 meters, that's why three different hubs had to be used in specific location to fulfill that requirement. Each hub had to be connected to computer with uplink USB cables. Since those hubs are quite far away from each other, extension cables were used. Each hub can also connect with up to 6 cameras with high speed USB cables, although we used three or four only ports. The camera and the documented required configuration for the setup is shown in Figure 12.
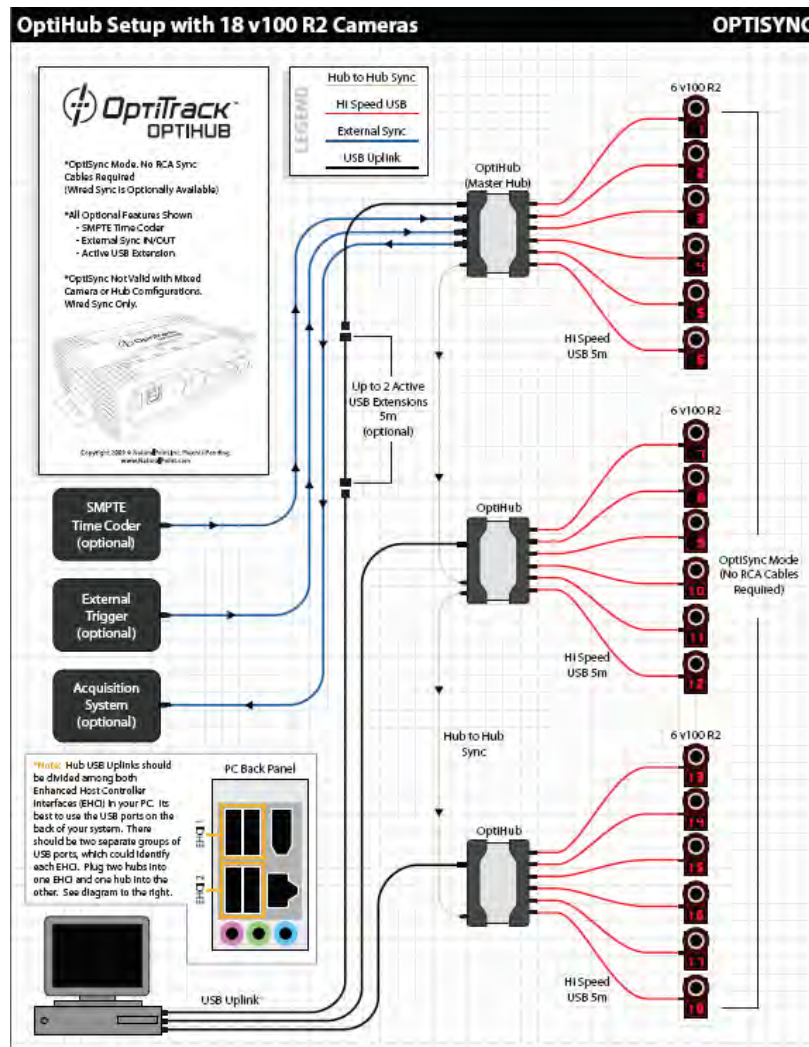
Figure 12.   Optihubs Setup with Optitrack Cameras. After [55].

23

After installing and power the whole system calibration of the cameras is needed. This procedure will be executed through the Optitrack Tracking Tools program, which is designed by the Optitrack Company and allows the operator to watch and adjust all the cameras' position into the laboratory and the view coming from each of them. This program is compatible with Quarc software, needed for controlling the vehicles. The calibration procedure is actually very easy and basically two tools are needed, a trident and a Γ-shape (gamma-shape) zero point instrument.

Firstly, the operator starts the "tracking tools" program. He has then to conduct two visual checks; checking that each camera is shown inside the program so that the connection is successful and that there is no reflection coming from the laboratory space. Then, he starts moving the trident in a figurate pattern (wanding process), whose edges have reflectors on the top, within the whole space trying to cover every possible view angle of each of the cameras He continues doing so until the tracking tools program indicates that all the cameras have calibrated successfully. Next, the zero point has to be adjusted with the Γ-shape instrument, where it shows the two axes, x and y. It is advisable that the vehicle has to be placed at this point in the beginning before the movement starts. Finally, since the calibration has completed, the calibration file has to be saved in order to be used in each Simulink vehicle model.

In order to localize the vehicles, this exact file has to be used inside the Quarc's software and especially the "Optitrack Point Cloud" Simulink block, a block that gives the position of the vehicle markers tracked by the Optitrack camera system. After that every model is ready to be used for any operation. Through that block you can always check how many markers are shown and if they are tracked successfully and in the event of an additional marker, thus unwanted reflection, to stop the operation and conduct another calibration after eliminating the issue. The cameras inside the laboratory are mounted accordingly and once adjusted, remain stable so that there is no need for continuous calibration.

F. QUANSER QUADROTOR Qball-X4

1. Introduction

The Qball-X4 quadrotor helicopter designed and constructed by the Quanser Company, is a rotary wing vehicle platform enclosed within a protective carbon fiber cage that is propelled by four motors fitted with 10-inch propellers. The particular design ensures safe operation in an indoor laboratory environment usually surrounded with various close range obstacles or other vehicles and it can be used in a wide variety of UAV research applications.

Qball-X4 utilizes Quanser's onboard avionics data acquisition card (DAQ), called HiQ that cooperates with an embedded computer by Gumstix in order to read the on-board sensors of the vehicle and drive the motors. The QBall-X4 is an open-architecture UAV, allowing operators to rapidly create and deploy unmanned vehicle controllers ranging from low-level flight dynamics stabilization to advanced multi-agent guidance, navigation and control algorithms. Other specifications of the QBall-X4 include:

- Diameter 0.7m – height 0.6m
- 2 LiPo batteries, 2500mAh, 3-cell
- 15 minute flight time per charge
- (740Kv) motors fitted with 10 inch propellers
- Protective cage made from carbon fiber enclosing the quadrotor
- USB camera up to 9 frames per sec color images
- Wireless communications capability

QUARC, Quanser's real-time control software, allows the operator to rapidly develop and test controllers on the host computer through a MATLAB Simulink interface, whereas these models can be generated and executed on the Gumstix embedded computer automatically in real time. At the same time operator can observe sensor measurements and tune the various parameters from the host ground station computer (PC or laptop).

The communication hierarchy for the operation of Qball-X4 is shown in Figure 13.
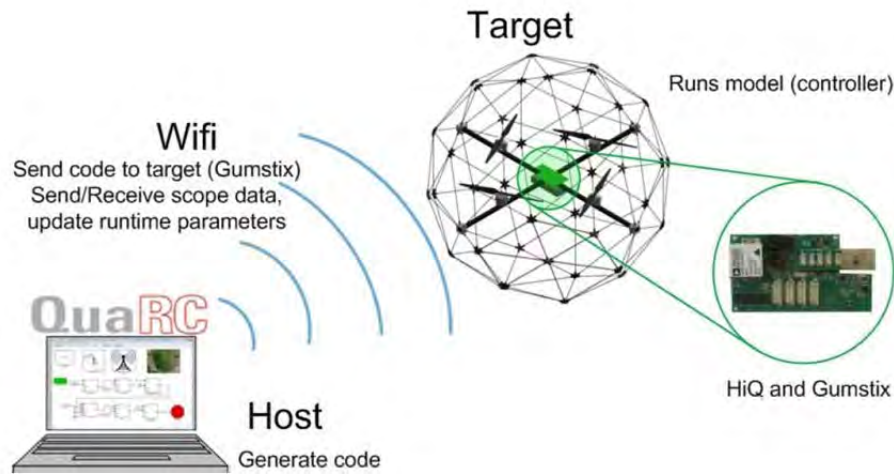
Figure 13.   Communication Hierarchy. After [51].

2.   X4 Diagram

The basic diagram of the Qball-X4 quadrotor with the axes and angles is shown in Figure 14, with the X axis aligned with the front of the vehicle. It is very common the tail or back of the vehicle, marked with colored tape, to point toward the operator and the positive X axis away from him when the vehicle operates.
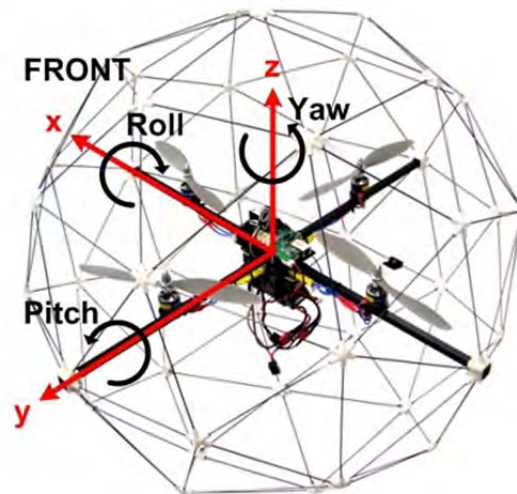
Figure 14.   Qball-X4 axes and sign convention. After [51].

3. Main Components

The main components of Qball-X4 are described below [51]:

a. *Qball-X4 Protective Cage and Frame.*

Qball-X4 quadrotor rests inside a protective frame (Figure 2.4) which is a crossbeam structure to which the Qball-X4 components are mounted. The Qball-X4's protective cage is a carbon fiber structure designed to protect the frame, motors, propellers, embedded control module (HiQ and Gumstix) and speed controllers during minor collisions, since it is not intended to withstand large impacts or drops from heights greater than 2 meters. That's why in order to move or lift it, someone has to carry it from the ends of the frame from both sides as it is shown in figure 15.

Figure 15.   Qball-X4 cage and frame. After [51]

b. *QUARC Data Acquisition Card (HiQ DAQ) - Gumstix Computer.*

The HiQ DAQ shown in Figure 2.6 is a high-resolution inertial measurement unit (IMU) and avionics input/output (I/O) data acquisition card that cooperating with the Gumstix embedded computer controls the vehicle by having inputs from the sensors on board and sending the motor commands. Each motor speed controller is connected in a specific order to one of the ten PWM servo output channels that are available on the HiQ, in order for the associated controllers to be operated. An optional daughterboard that contains additional I/O such as receiver or sonar inputs or a TTL serial input used for a GPS receiver. The standard servo output channels associated with every motor and the most common channel associated with sonar in the daughterboard are shown in Table 3.
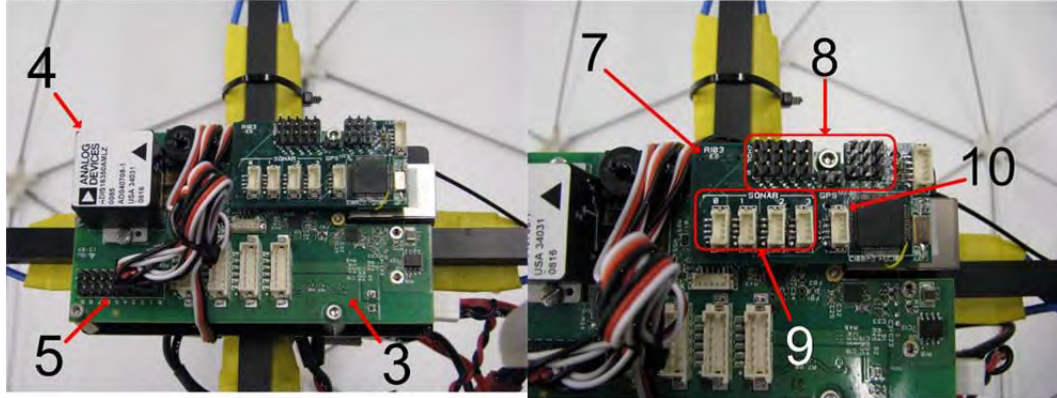
Figure 7: HiQ DAQ-HiQ daughterboard with optional receiver inputs. After [51].

Main Board	
Motor	Servo Output Channel
Back	0
Front	1
Left	2
Right	3
Daughterboard	
Sonar	0

Table 3.　　Servo Output Channel
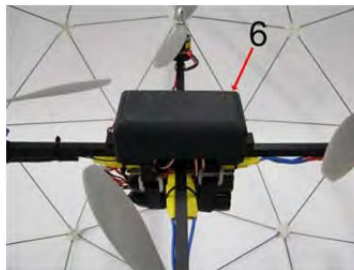
Figure 16.　　HiQ cover. After [51].

c.   *Qball-X4 Power - LiPo Batteries – Switches – Connectors*

The power to the Quadrotor (HiQ and motors) is provided by two 3-cell 2500mAh LiPo batteries (#14 in Figure 17) which should be secured tightly in a vertical position with velcro straps on the bottom side of the frame. After connecting the batteries with the battery connectors, the power is turned on by using two switches (one for each battery) (#12 in Figure 17). Because of the fact that these batteries will be damaged and turned to no use if they are discharged below 10 V, they should be fully charged when they reach 10.6 V or less.
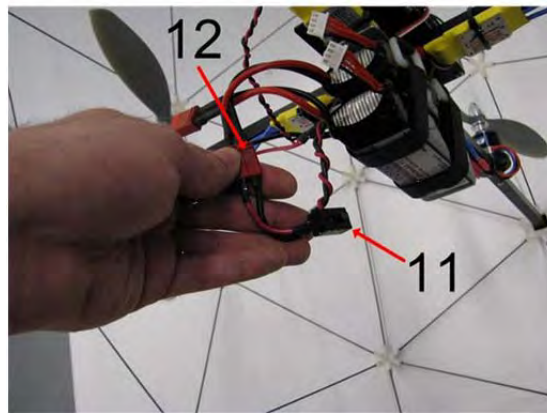
Figure 17.   Battery switch and connector. After [51].

d.   *Motors, Propellers (10x4.7) and Speed controllers (ESCs)*

The motors used in the quadrotor are four E-Flite Park 400 (740 Kv) motors (Figure 18) fitted with paired counter-rotating APC 10x4.7 propellers [4] (#16 in Figure 18), mounted and connected with the four electronic speed controllers (ESCs) [5], along the X and Y axes of the frame.

The motors and propellers are configured so that the front and back motors spin clockwise and the left and right motors spin counterclockwise. The ESCs receive commands from the HiQ in the form of PWM outputs from 1ms (minimum throttle) to 2ms (maximum throttle) or can be configured to set the throttle range, but always with initial PWM outputs to the minimum throttle value of 0.05.
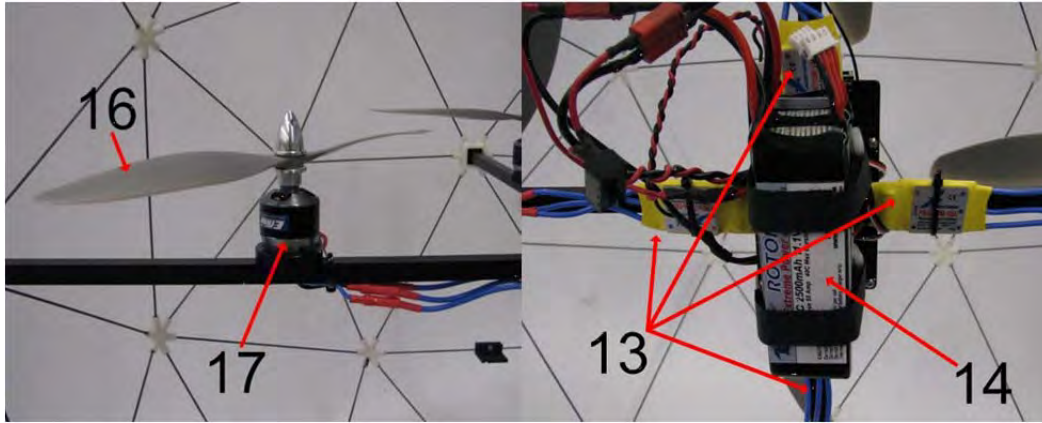
Figure 18.   Motor and propeller–ESCs and batteries. After [51].
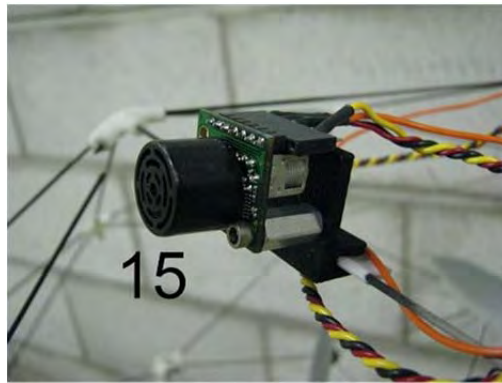
Figure 19.   Optional sonar and sonar mount. After [51].

ID #	Description	ID #	Description
1	Qball-X4 protective cage	10	GPS serial input
2	Qball-X4 frame	11	Battery switch
3	HiQ DAQ with Gumstix	12	Battery connector
4	HiQ inertial measurement unit	13	Speed controllers (ESCs)
5	HiQ servo PWM outputs	14	LiPo batteries
6	HiQ cover	15	Optional sonar
7	HiQ daughterboard with optional receiver	16	Propeller (10x4.7)
8	Receiver inputs	17	Motor
9	Sonar inputs		

Table 4.   Description of Qball's main components. After [51].

4.   System Set up

a.   *Wireless Connection*

The quadrotor communicates with the host computer and the ground robot by utilizing an ad-hoc peer-to-peer wireless TCP/IP connection through a USB wireless adapter. The network established is called GSAH and is an unsecured one, with the following settings / properties:

IP address: 182.168.1.xxx (any unused number)
Subnet mask: 255.255.255.0
Default gateway: 182.168.1.xxx (same as IP)
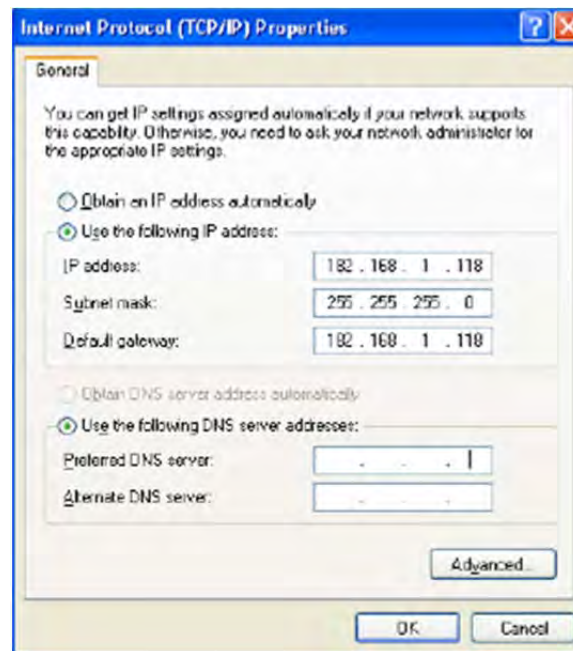
Figure 20.   Wireless USB adapter settings

b.   *Qball-X4 Vehicle Setup*

After checking that the motors, propellers are firmly secured to the frame regularly (after every 2 hours of flight), the batteries must be installed and connected to the battery connectors in the way described above and is shown in Figure 21.
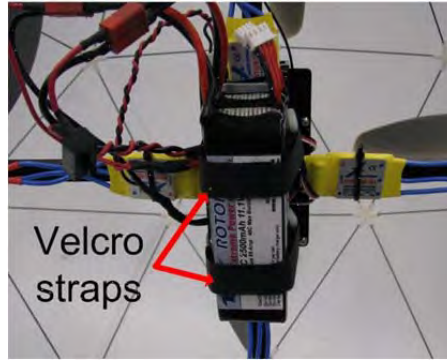
Figure 21.   Batteries secured with velcro straps. After [51].

To power on the Qball-X4, the two power switches connected to the battery cables (#11 in Figure 8) must be turned on, initiating the Gumstix wireless module one minute after. Then if connected to the GSAH ad-hoc network on the host PC, the quadrotor is ready to be navigated.

c.   Quanser Real-Time Control (QuaRC) Software Configuration

After installing QuaRQ software, a new item will be added in Simulink menu. Some configurations must be done to be able to run any QuaRC model on the target vehicle.

First of all, the target's Gumstix IP address must be specified. So, the operator has to setup the default target IP address for all targets, inside theQuaRC menu through the Preferences option (e.g., "tcpip://182.168.1.200:17001").

Moreover, "External" simulation mode must be selected instead of "Normal" to run the model on the target machine (Gumstix), otherwise only a simulation will be run on the host machine.

Finally, the building of the model (QuaRC/Build) only remains in order to begin the code generation and compiling steps on the target vehicle. QUARC console will show when the compilation will be finished (usually this process takes some minutes).

d.　　Qball-X4 Sensors Associated with QuaRC Blocks

QuaRC's open-architecture hardware and extensive Simulink block set provides users with powerful controls development tools. Several of these blocks of the QuaRQ software have to be used in order to read the sensors attached in the quadrotor and provided the data in Simulink models. The most important are the Hardware-In-the-Loop (HIL) block for communication with HiQ board. The HIL Initialize block for initialization the HiQ and setup the I/O parameters and the HIL Read Write block for reading and writing from the HiQ to the model.

The quadrotor's commands for the four motors are associated with the PWM outputs 0 to 3. The range of the PWM output values is between 0.05 (zero throttle) which corresponds to a 1ms pulse (5% of a 20ms duty cycle) and 0.10 (full throttle), equivalent to 2ms pulse (10% of a 20ms duty cycle).

To control the flight of the quadrotor several measurements are required from the IMU. The input from the magnetometer, functioning as a digital compass is used to measure the heading (corresponding yaw angle) while the 3-axis gyroscope and accelerometer inputs are used to measure the quadrotor dynamics and orientation (roll, pitch and yaw angles).

As already noted, the LiPo batteries should be charged at 10.6V, otherwise the batteries will be destroyed. The block "Show Message on Host" allows the operator to check the battery capacity by displaying a low battery warning message on the host PC. The operating capacity input measures the battery value as a percentage (0-1%) of the quadrotor's input voltage range from the minimum value of 10V to the maximum one of 20V (10.6V is equivalent to 0.06 or 6%).

G.   QUANSER QBOT GROUND VEHICLE

1.   Introduction

The Quanser Qbot (Figure 22) is an autonomous ground robot vehicle consists of the iRobot Create platform widely used in robot applications with four infrared and three sonar sensors and a Logitech Quickcam Pro 9000 USB camera mounted on the vehicle, and the embedded Quanser Controller Module (QCM) utilizing a Gumstix computer so as to run QuaRC, Quanser's real-time control software and the Qbot data acquisition card (DAC). So, in other words, Quanser has taken the iRobot Create platform and augmented its sensor capabilities by adding [54]:

- 8 PWM outputs for servo motors
- 7 reconfigurable digital I/O, plus 1 digital output LED
- 7 analog inputs, 12-bit, +5V inputs, resolution 6.2 mV
- 5 infra-red (IR) sensors up to 150cm
- 3 sonar sensors 15cm to 6.45m, 1-inch resolution
- 3-axis magnetometer, resolution of 0.77 milli-Gauss
- USB camera up to 9fps color images
- Wireless communications
-
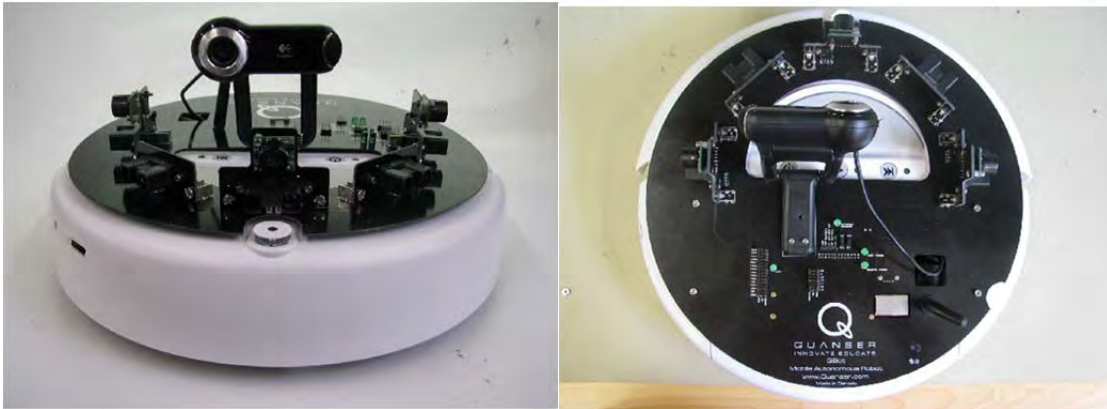
Figure 22.   The Quanser Qbot – front (Left) and top (Right) view. After [54].

The QCM interface is a MATLAB Simulink with QuaRC as in the quadrotor. The Qbot is accessible through three different block sets: the Roomba block set to drive the

vehicle, the HIL block set to read from sensors and/or write to servo outputs, and finally the OpenCV block set to access the camera. The controllers are developed in Simulink with QuaRC on the host computer, and these models are downloaded and compiled into executables on the target [54] seamlessly. A diagram of this configuration is shown in Figure 23 and the characteristics of the Qbot vehicle is shown in Table 5.
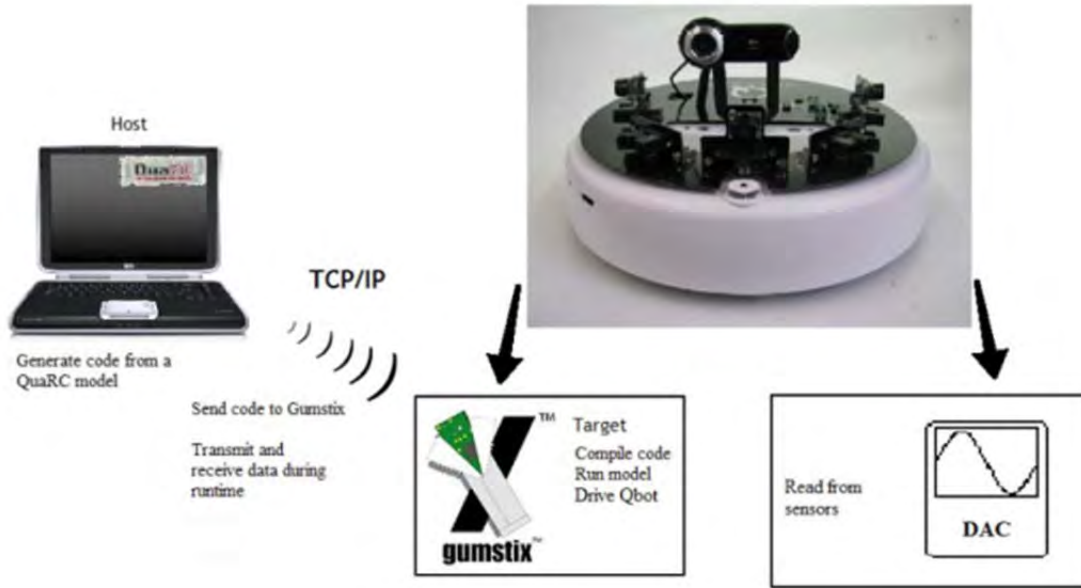
Figure 23.   Communication Hierarchy. After [54].

Qbot Specifications			
Symbol	**Description**	**Value**	**Unit**
D	Diameter	0.34	m
H	Height (With Camera Attachment)	0.19	m
Vmax	Maximum speed	0.5	m/sec
M	Total mass	2.92	Kg

Table 5.   Qbot Model Parameters and System Specifications. From [54]

2. Main Components Description

The main components of the Qbot are described in the next section [54].

a. *The iRobot Create®*

The Qbot uses an iRobot Create® frame (Figure 24). The Qbot body frame axes are the Quanser standard, where the x-axis is in the forward direction, the y-axis is to the left, and the z-axis is up. The iRobot Create® has a bumper sensor and an omni-directional infrared receiver. The QCM is capable to read these sensors and receiving their data. Two differential drive wheels drive the vehicle of diameter of 34 cm and height (without camera attachment) of 7 cm.
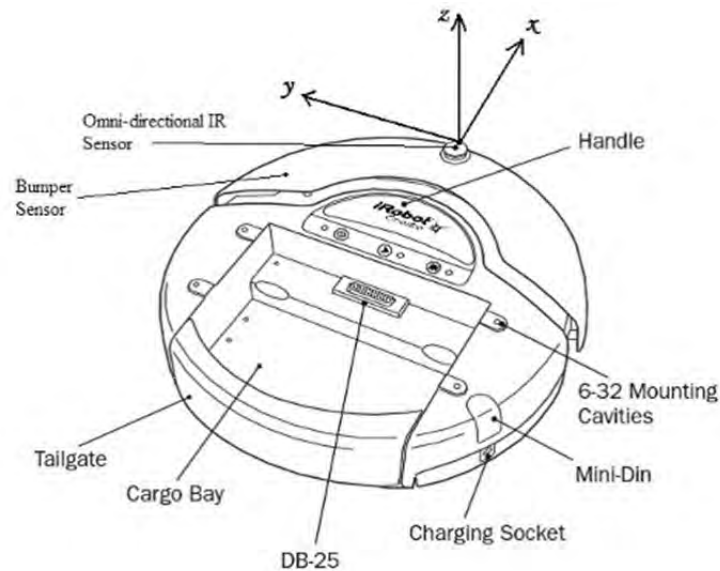
Figure 24.   Anatomy of Qbot, showing various components and body axes. After [54].

The Qbot is turned on by pressing the power button. Built-in demos for the vehicle are also available through the Play and Advance buttons (Figure25).
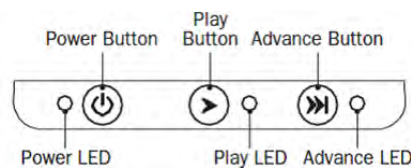
Figure 25.   Buttons on the Qbot Frame. After [54].

36

b. Qbot DAC

The Qbot DAC is a data acquisition board, located underneath the black cover of the Qbot, existed for receiving analog inputs and sonar inputs or any other input from different optional sensors. It is also capable of writing PWM outputs for possible servo actuators.

c. Gumstix Computer

The Gumstix is a small-scale, fully functional, open source computer at where the MATLAB/ Simulink models are directly downloaded, compiled, and executed through the QuaRC software. The Gumstix motherboard is connected directly to the Qbot DAC. Wifi attachment board to allow wireless connection between the target Gumstix and the host computer and/or other vehicles is also available.

d. The Printed Circuit Board (PCB)

The wiring and circuitry for the Qbot in the printed circuit board (PCB) that is located on the black cover of the Qbot over the Gumstix computer and the DAC. The sensors and the webcam are also mounted on the PCB. Figure 26 shows the accessible pins for the user. In particular, the DIO, PWM output, and analog input pins have been labeled for clarity.
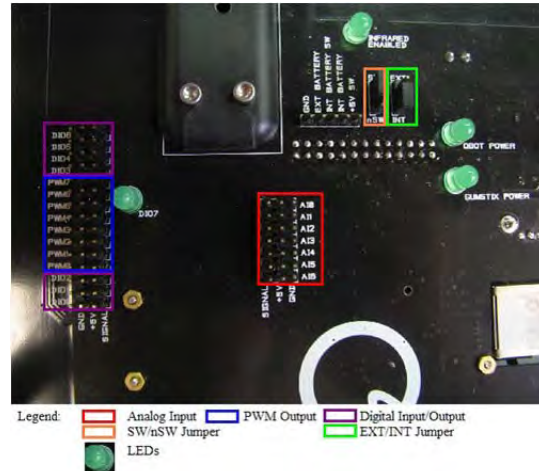
Figure 26.   Qbot PCB showing available pins for the PWM output, Digital Input/Output and Analog Input pins, and jumpers for INT/EXT power. After [54].

e.   *Digital Input/Output Pins (DIO #)*

The DIO channels (0 to 6) are set as inputs by default. With the HIL Initialize block, the operator can configure the DIO channels either as inputs or outputs (not both). There is also a fixed output indicated by a LED labeled DIO7 associated with a final digital channel (7).

f.   *SW/nSW and INT/EXT Jumpers*

The operator can switch between internal power from the iRobot Create battery (INT) and an external battery power supply (EXT) with the power source jumper INT/EXT jumper. To power the Qbot, the jumper should be placed in the INT position and then the SW/nSW jumper will indicate if the iRobot Create must be switched on (SW) for the Qbot to receive power, or whether the Qbot should always draw power even when the iRobot Create is off (nSW).

g.   *USB Camera*

The camera is mounted on top of the Qbot (Figure 27). The QuaRC block set that uses the Open Source Computer Vision library giving the opportunity to the operator to capture and display images in real time, process them, and even save them. The image Logitech Quickcam Pro 9000 USB resolution though is low.

Figure 27.   Logitech Quickcam Pro 9000 USB Camera. After [54].

h.   Battery

The Qbot is powered by the Advanced Power System (APS) Battery (Figure 28-Left) provided by iRobot and placed in specific location under the Qbot (Figure 28-Right). If the battery is fully charged, is able to last continuously for about 2 hours. The power level of the battery is designated by the power light (green for fully charged / red for discharged batteries) on the Qbot platform. The battery takes about 3 hours to fully charge.
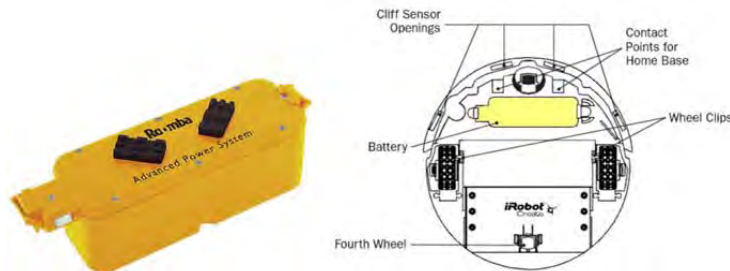
Figure 28.   The Qbot the Advanced Power System Battery (Left picture) - Battery location highlighted in the Bottom view of the Qbot. After [54].

i.   Infrared Sensors - Sonar Sensors

Qbot has five SHARP 2Y0A02 low cost infrared range sensors (20-150 cm) (Figure 29-Left) and three MaxSonar-EZ0, very short to long range detection and ranging, sonar sensors (Figure 29-Right), that are connected to the analog input channels of the Qbot DAC. Readings of those sensors can be taken through the HIL Read Write block of the QUARC software. The sonar provides detection range between 0 and 254-inches (6.45 meters) while the information provided covers range from 6-inches out to 254-inches with resolution of 1 inch.

Figure 29.   SHARP GP2D12 IR Sensor (Left) - LV-MaxSonar-EZ0 Sonar Range Finder (Right). After [54].

3.   System Setup

a.   Setting up the Qbot

To set up the Qbot the operator has to follow two steps:

(1)   Insert the battery under the Qbot to the designated area.

(2)   Press the power button to turn on both the robot and Qbot DAC/Gumstix.

b.   Establishing Wireless Connection

The Qbot communicates with the host computer and the quadoror by utilizing an ad-hoc peer-to-peer wireless TCP/IP connection through a USB wireless adapter. The network established is the same GSAH one, with the settings / properties described before for the quadrotor.

c.   Quanser Real-Time Control (QuaRC) Configuration

Having installed QUARQ software, the same configurations with the quadrotor have to be done for the Qbot as well, so as to be able to run any QUARC model on the target vehicle.

First of all, the target's Gumstix IP address must be specified. So, the operator has to setup the default target IP address for all targets, inside the QUARC menu through the Preferences option (e.g., "tcpip://182.168.1.200:17001").

Moreover, "External" simulation mode must be selected instead of "Normal", so as to run the model on the target machine (Gumstix), otherwise only a simulation will be run on the host machine.

Next, the building of the model (QUARC/Build) only remains in order to begin the code generation and compiling steps on the target vehicle. QUARC console will show when the compilation will be finished (usually this process takes some minutes).

d.   *Qbot Sensors Associated with QuaRC Blocks*

Several blocks of the QUARQ software have to be used in order to read the sensors attached in the Qbot and provided the data in Simulink models. The most important are described below [54]:

(1)   The Roomba Initialize block located in the Simulink Library Browser, under QuaRC Targets / Devices / Third-Party / iRobot / Roomba / Interfacing, is required for the Gumstix computer to communicate with the Qbot. The operator must change the local host from the default 2 to 1.

(2)   The HIL Initialize block located in the Simulink Library Browser, under QuaRC Targets / Data Acquisition / Generic / Configuration, is required to communicate with the Qbot DAC. When model runs, the board type "qbot" must be selected in order to target the Qbot.

(3)   The HIL Read Write block located in the Simulink Library Browser, under QuaRC Targets / Data Acquisition / Generic / Immediate I/O is required to read infrared sensor (analog) and sonar measurements from Qbot.

(4)   The blocks located in the Simulink Library Browser, under QuaRC Targets Beta / Image Processing / Open Source Computer Vision are required to use the on-board Logitech camera for image processing.

H.   TOWARD UNIFIED CONTROL ARCHITECTURE

From the standpoint of controls QBall and QBot are shown in Figures 30 and 31, respectively. On the left the control inputs that affect vehicles dynamics and kinematics are introduced. On the right the available output signals are shown. The position feedback is provided by the motion capture cameras. While both systems are capable to operate autonomously, the communication with the control station is provided via a wireless connection. Multiple systems can talk to each other directly as well.

Quadrotor QBall-X4 Architecture

Figure 30.   Ground Robot QBot Architecture
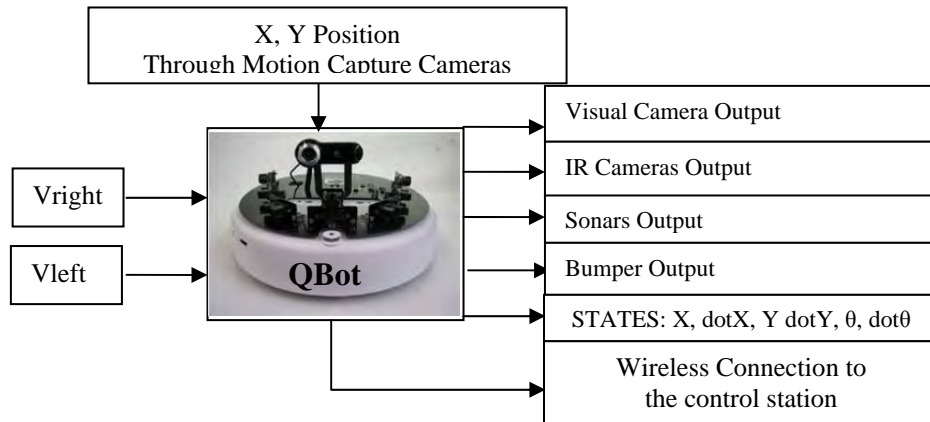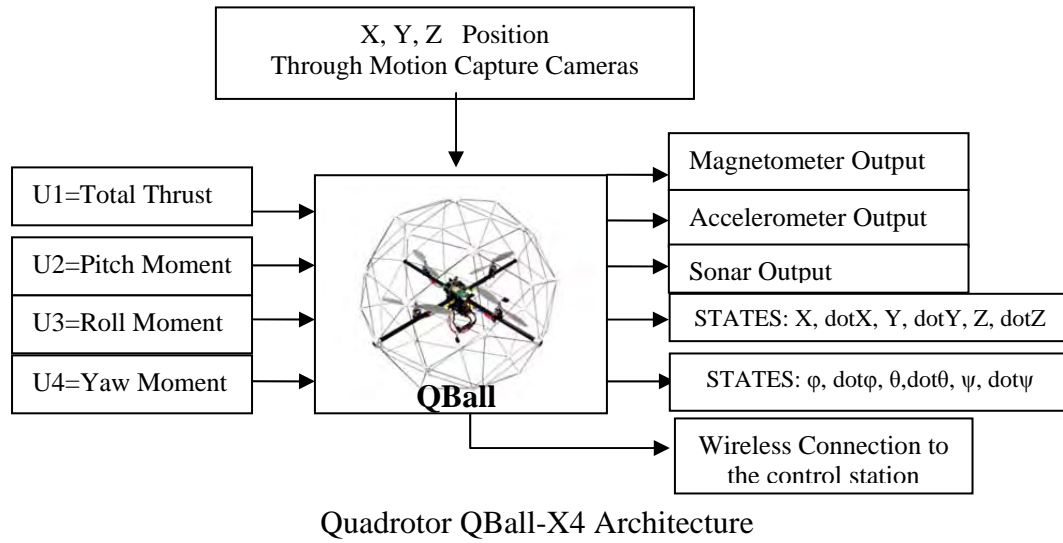
Table 6 shows sensors available on both vehicles side by side.

Vehicles / Sensors	QBall – X4	QBot
IR Camera	-	+
Visual Camera	-	+
Sonar	+	+
Bumper	-	+
Accelerometer	+	-
Magnetometer	+	-

Table 6.   Comparison of the two vehicles sensors

QBall can also be equipped with the IR/EO camera. In terms of performance,

- QBall has a large velocity compare o that of QBot which opportunity leads to the fast accomplishment of mission.

- The higher height of QBall operation results in larger view of the field of operation and better operation coverage.

- QBot has a larger payload / sensor capability and therefore better obstacle avoidance and identification of the target capability.

Hence these two vehicles complement each other, and that's why their cooperation could improve the chances of mission success.

THIS PAGE INTENTIONALLY LEFT BLANK

III. MODELLING OF QUADROTOR AND GROUND ROBOT

In this chapter, the dynamic and kinematics modeling of the quadrotor and the ground robot will be presented to the extent of deriving the final equations describing the vehicles and that they will be used for the design of the controllers. The general dynamic model of the quadrotor has been presented in many papers and can be derived through two different approaches, the Euler – Newton and the Euler - Lagrange formalisms, see [1]. The modeling of a ground robot is modeled with the forward and the inverse kinematics; see [56] and [57].

A. MODELLING OF QUADROTOR QBALL-X4

In order to start modeling the quadrotor, the coordinate frames that were used, have to be presented, described and defined.

1. Coordinate Frames

a. Earth-Fixed Inertial Frame U

The inertial frame is established by the direction of the Earth's rotation. The coordinate frame to be used is the North-East-Up, with the origin O_U generally at an arbitrary ground point, here chosen to be at the quadrotor take-off point. As shown in figure 32, the unit vector 'x' points toward North, the unit vector 'y' toward East and 'z' toward the opposite direction of the center of the Earth.
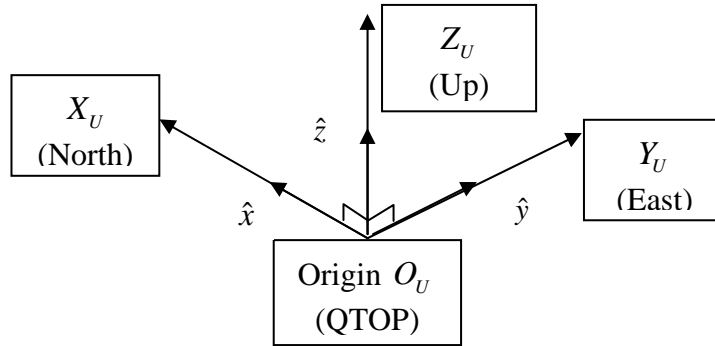
Figure 31.   Earth Fixed Inertial Frame U

b. *Body- Fixed Frame B*

The body frame is rigidly attached to and defined within the vehicle and exists to specify quadrotor's orientation. The definition for this frame has the x-axis along the two opposing propellers axis, the y axis pointed along the other two opposing propellers and the z-axis upward, forming the right hand set. This frame is shown below in figure 33 and Roll, Pitch and Yaw are defined as the angles of rotation φ, θ and ψ about x, y and z axis, respectively. The origin of the body frame is at the center of the mass of the quadrotor.
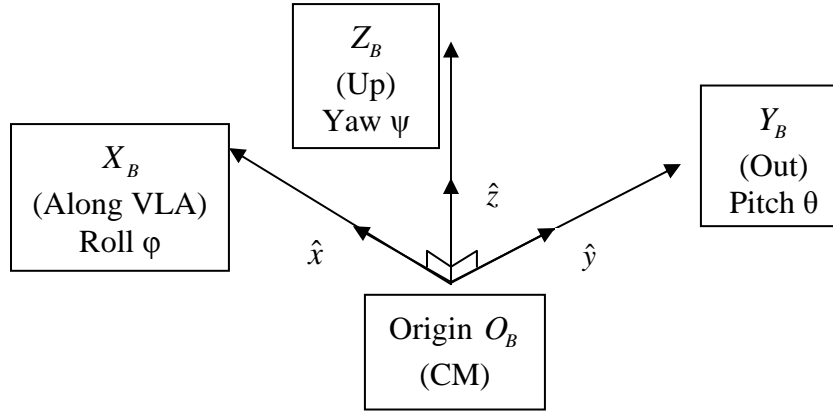
Figure 32.   Body-fixed Frame B

2.   Modeling Assumptions

At this point several reasonable assumptions concerning the modeling of the quadrotor must also be made:

- The Earth is flat, non-rotating, and an approximate inertial reference frame.

- The acceleration of gravity is constant and perpendicular to the surface of the Earth.

- The design is symmetrical with respect to the axis.

- The quadrotor body as well as the propellers will be treated as rigid body, so that the Newton- Euler Formulas can be used.

- Since, it is an indoor experiment, and the speed is considered to be low, the air friction will be ignored, gyroscopic effects and the aerodynamic torques can be cancelled.

- No swash plate exists for each rotor.

3. Vehicle State Variables

The quadrotor QBALL X-4 is an unmanned helicopter with four rotors combined with a cross scheme. The quadrotor generates its lift by these four rotors. The two opposing rotors form one pair, where the first pair (#1 and #3) is set on the *x*-axis and is rotated counter clockwise while the second one (#2 and #4) is set on the *y*-axis, rotated counterclockwise as shown in Figure 34.
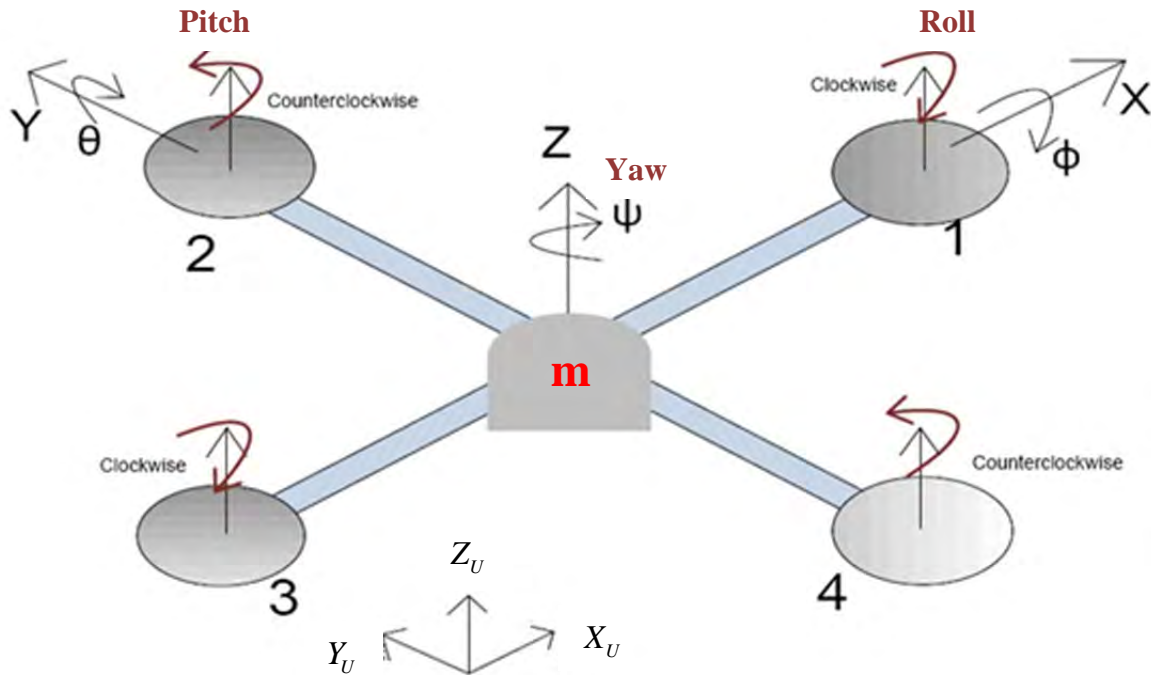
Figure 33.   Quadrotor Configuration Scheme

The earth-fixed inertial frame (*Xu*, *Yu*, *Zu*) shown, specify the location of the vehicle, while the body frame (*Xb*, *Yb*, *Zb*) specify the vehicle orientation, together with angles of rotation, roll (ϕ), pith (θ), and yaw (ψ).

Let the mass of the quadrotor be m that represents the whole structure mass of the quadrotor, as it stated in the assumption (d) in the previous section.

Let ^{B}V the velocity vector of the vehicle and $^{B}\omega$ the rate of change of the angle (angular velocity) in body frame, respectively, ^{u}x the position of the vehicle in the inertial system

$$\overrightarrow{^{B}V} = u\vec{i} + v\vec{j} + w\vec{k} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{1}$$

$$^{B}\vec{\omega} = p\vec{i} + q\vec{j} + r\vec{k} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2}$$

$$^{u}\vec{x} = x\vec{i} + y\vec{j} + z\vec{k} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{3}$$

Also, the Euler angles of the vehicle which rotates around the XYZ Body frame axis are represented as

$$\Lambda = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} \tag{4}$$

4. Transformation Matrices

In order to transform the state vectors from {U} frame to the {B} frame, a transformation matrix must be used. This matrix according to the convention of rotate first around x-axis, then around y axis and finally by the z axis as described in [1] can be found by the following equation:

$$^{B}_{u}R = R_{x,\phi} * R_{y,\theta} * R_{z,\psi} \tag{5}$$

where $R_{x,\phi}, R_{y,\theta}, R_{z,\psi}$ are the rotation matrices around each axis, respectively, as follows:

Rotation around x-axis: $R_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}$ (6)

with yaw angle $\psi \in \left(-\dfrac{\pi}{2}, \dfrac{\pi}{2} \right)$

Rotation around y-axis: $R_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$ (7)

with pitch angle $\theta \in \left(-\dfrac{\pi}{2}, \dfrac{\pi}{2} \right)$

Rotation around z-axis: $R_{z,\psi} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$ (8)

with roll angle $\phi \in \left(-\dfrac{\pi}{2}, \dfrac{\pi}{2} \right)$

So, if we use the Eq. 6, 7 and 8 in Eq. 3.5, the final form of the transformation matrix can be derived:

$$ _u^B R = R_{x,\phi} * R_{y,\theta} * R_{z,\psi} = $$

$$ \begin{bmatrix} \cos\theta\cos\psi & \sin\psi\cos\theta & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & -\sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{bmatrix} $$ (9)

The transformation matrix from {B} to the {U} coordinate frame is:

$$ _B^u R = \begin{bmatrix} \cos\theta\cos\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\sin\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ \sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} $$ (10)

5. Kinematic Equations

Having defined the transformation matrices, we can now find the velocity of the vehicle in the inertial frame instead of the Body coordinate frame and it can be given by:

$$^u\dot{x} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = {}_B^u R\,{}^B V = {}_B^u R \begin{bmatrix} u \\ v \\ w \end{bmatrix} =$$

$$= \begin{bmatrix} (\cos\theta\cos\psi)u + (-\cos\varphi\sin\psi + \sin\varphi\sin\theta\sin\psi)v + (sin\varphi\sin\psi + \cos\varphi\sin\theta\cos\psi)w \\ (\cos\theta\sin\psi)u + (\cos\varphi\cos\psi + \sin\varphi\sin\theta\sin\psi)v + (-\sin\varphi\cos\psi + \cos\varphi\sin\theta\sin\psi)w \\ (\sin\theta)u + (\sin\varphi\cos\theta)v + (\cos\varphi\cos\theta)w \end{bmatrix} (3.11)$$

Moreover, the rate of change of the Euler angles of the vehicle as stated in [1] can be found by:

$$\dot{\Lambda} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = Q^{-1}(\Lambda)\,{}^B\omega \tag{12}$$

where the matrix Q(Λ) and its inverse one, $Q^{-1}(\Lambda)$ are given by the equation:

$$Q(\Lambda) = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{13}$$

$$Q^{-1}(\Lambda) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \tag{14}$$

So, using the Eq. 2 in Eq. 12 the following equation can be derived:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{15}$$

The last equation and Eq. 11 are the **kinematic equations** of the quadrotor.

6. Dynamic Equations

According to **Newton's second law** in the {U} orientation frame, the following equation represents the applied force ^{u}F and the acceleration of the vehicle $^{u}\ddot{x}$ in the {U} frame:

$$\sum {}^{u}F = \begin{bmatrix} F_{x} \\ F_{y} \\ F_{z} \end{bmatrix} = \frac{d}{dt}(mV) = m\frac{d}{dt}\vec{V} = m\,{}^{u}\ddot{x} = m\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \tag{16}$$

If we multiply by both sides the transformation matrix ${}_{u}^{B}R$ of Eq. 9, Eq. 16 will be

transformed to:
$$\sum {}_{u}^{B}R\,{}^{u}F = \sum {}^{B}F = {}_{u}^{B}Rm\,{}^{u}\ddot{x} = m\,{}_{u}^{B}R\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \tag{17}$$

$$\Rightarrow \sum {}^{B}F = m\,{}_{u}^{B}R\frac{d}{dt}({}^{u}\dot{x}) \tag{18}$$

Continuing the math:

$$\sum {}^{B}F = m\,{}_{u}^{B}R\frac{d}{dt}({}_{B}^{u}R\,{}^{B}V) = m\,{}_{u}^{B}R({}_{B}^{u}R\,{}^{B}\dot{V} + {}_{B}^{u}\dot{R}\,{}^{B}V) = m({}_{u}^{B}R\,{}_{B}^{u}R\,{}^{B}\dot{V} + {}_{u}^{B}R\,{}_{B}^{u}\dot{R}\,{}^{B}V) \Rightarrow$$

$$\sum {}^{B}F = m({}^{B}\dot{V} + {}_{u}^{B}R\,{}_{B}^{u}\dot{R}\,{}^{B}V) = m({}^{B}\dot{V} + {}_{u}^{B}R[{}_{B}^{u}RS({}^{B}\omega)] \times {}^{B}V) \Rightarrow$$

$$\sum {}^{B}F = m({}^{B}\dot{V} + S({}^{B}\omega) \times {}^{B}V)) \tag{19}$$

Since ${}_{u}^{B}\dot{R} = {}_{B}^{u}RS({}^{B}\omega)$, where $S({}^{B}\omega)$ is a skew symmetric matrix. The term $S({}^{B}\omega) \times {}^{B}V$ is the Coriolis term and can be written as:

$$S({}^{B}\omega) \times \overrightarrow{V^{B}} = {}^{B}\vec{\omega} \times \vec{V}^{B} = \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix} \tag{20}$$

If we substitute the derivative of Eq. 1 and the Eq. 20 into Eq. 19 and rearrange the terms, then:

$$\sum {}^{B}F = m\left(\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix}\right) \Rightarrow \frac{1}{m}\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} + \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix} \Rightarrow$$

$$\Rightarrow \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{1}{m}\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} - \begin{bmatrix} qw - rv \\ ru - pw \\ pv - qu \end{bmatrix} \tag{21}$$

where F_x, F_y, F_z are the forces in the body coordinate frame in each axis, u, v, w and $\dot{u}, \dot{v}, \dot{w}$ are the velocity and the acceleration components in the body frame, respectively, in each axis and p, q, r are the angular velocity components in the body frame in each axis.

Using the Euler's Law, the momentum ${}^{u}M_0$ in the inertia frame will be given by:

$$\sum {}^{u}M_0 = \frac{d}{dt}({}^{u}\vec{A}) = \frac{d}{dt}({}_{B}^{u}RJ_B \, {}^{B}\vec{\omega}) =$$

$${}_{B}^{u}\dot{R}J_B \, {}^{B}\vec{\omega} + {}_{B}^{u}RJ_B \, {}^{B}\dot{\vec{\omega}} + {}_{B}^{u}R\dot{J_B} \, {}^{B}\vec{\omega}^{\,0} = {}_{B}^{u}R({}^{B}\vec{\omega} \times J_B \, {}^{B}\vec{\omega}) + {}_{B}^{u}RJ_B \, {}^{B}\dot{\vec{\omega}} \Rightarrow$$

$$\Rightarrow \sum {}_{u}^{B}R \, {}^{u}M_0 = {}_{u}^{B}R[{}_{B}^{u}R({}^{B}\vec{\omega} \times J_B \, {}^{B}\vec{\omega})] + {}_{u}^{B}R[{}_{B}^{u}RJ_B \, {}^{B}\dot{\vec{\omega}}] \Rightarrow$$

$$\Rightarrow \sum {}^{B}M_0 = \begin{bmatrix} L \\ M \\ N \end{bmatrix} = ({}^{B}\vec{\omega} \times J_B \, {}^{B}\vec{\omega}) + J_B \, {}^{B}\dot{\vec{\omega}} \tag{22}$$

where L, M, and N rolling, pitching and yawing moments, respectively, in the body coordinate system, ${}^{B}\vec{A}$ the angular momentum in the Body frame and J_B is the Inertia Tensor of the quadrotor in the B coordination frame ($J_B \in \mathbb{R}^{3 \times 3} > 0$):

$$J_B = \begin{pmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{yz} & J_{yy} & J_{yz} \\ J_{zx} & J_{zy} & J_{zz} \end{pmatrix} \tag{23}$$

Then, the angular acceleration will be given by the equation:

$$^{B}\dot{\vec{\omega}} = J_{B}^{-1}[\sum{}^{B}M_{0} - (^{B}\vec{\omega} \times J_{B}{}^{B}\vec{\omega})] \tag{24}$$

where

$$J_{B}^{-1} = \begin{bmatrix} \dfrac{1}{J_{xx}} & 0 & 0 \\ 0 & \dfrac{1}{J_{yy}} & 0 \\ 0 & 0 & \dfrac{1}{J_{zz}} \end{bmatrix} \tag{25}$$

since $J_{xy} = J_{yz} = J_{xz} = J_{zx} = J_{yx} = J_{zy} = 0$ because the quadrotor is symmetric about all three axes.

If we take the derivative of the angular velocity in Eq. 2:

$$^{B}\dot{\vec{\omega}} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{26}$$

And combine Eq. 2 with Eq. 23 to take the external product, we will have:

$$^{B}\vec{\omega} \times J_{B}{}^{B}\vec{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} J_{xx}p - J_{xy}q - J_{xz}r \\ -J_{yx}p + J_{yy}q - J_{yz}r \\ -J_{zx}p - J_{zy}q + J_{zz}r \end{bmatrix} = \\ \begin{bmatrix} -r(-J_{yx}p + J_{yy}q - J_{yz}r) + q(-J_{zx}p - J_{zy}q + J_{zz}r) \\ r(J_{xx}p - J_{xy}q - J_{xz}r) - p(-J_{zx}p - J_{zy}q + J_{zz}r) \\ -q(J_{xx}p - J_{xy}q - J_{xz}r) + p(-J_{yx}p + J_{yy}q - J_{yz}r) \end{bmatrix} \tag{27}$$

Finally, if we substitute Eq. 3.31 and Eq.3.33 in equation 3.30 we will have:

$$^{B}\dot{\vec{\omega}} = \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = J_{B}^{-1}(\begin{bmatrix} L \\ M \\ N \end{bmatrix} - \begin{bmatrix} -r(-J_{yx}p + J_{yy}q - J_{yz}r) + q(-J_{zx}p - J_{zy}q + J_{zz}r) \\ r(J_{xx}p - J_{xy}q - J_{xz}r) - p(-J_{zx}p - J_{zy}q + J_{zz}r) \\ -q(J_{xx}p - J_{xy}q - J_{xz}r) + p(-J_{yx}p + J_{yy}q - J_{yz}r) \end{bmatrix}) \Rightarrow$$

$$\Rightarrow \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dfrac{L}{J_{xx}} \\ \dfrac{M}{J_{yy}} \\ \dfrac{N}{J_{zz}} \end{bmatrix} - \begin{bmatrix} \dfrac{qr}{J_{xx}}(J_{zz} - J_{yy}) \\ \dfrac{pr}{J_{yy}}(J_{xx} - J_{zz}) \\ \dfrac{pq}{J_{zz}}(J_{yy} - J_{xx}) \end{bmatrix} \qquad (28)$$

The last equation together and the Eq. 21 are the **dynamic equations** of the quadrotor. Combining the kinematic and the dynamic equations of the quadrotor you end up with the nonlinear differential equations of motion of a rigid body with six degrees of freedom.

7.   Forces Calculation

a.   *Gravity Force*

Gravity F_G is the main force acting on the quadrotor in the earth fixed frame along z-axis toward the ground, which is given by the total mass of the quadrotor multiplied by the acceleration of the gravity as follows:

$$^u F_G = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \qquad (29)$$

where m is the total mass of the vehicle, g is the acceleration of the gravity. Consequently, Eq. 29 will be transformed in the body coordination frame using the transformation matrix $^B_u R$ from Eq. 9:

$$^B F_G = {}^B_u R \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = \begin{bmatrix} -mg \sin\theta \\ mg \sin\varphi \cos\theta \\ mg \cos\varphi \cos\theta \end{bmatrix} = \begin{bmatrix} F_{Gx} \\ F_{Gy} \\ F_{Gz} \end{bmatrix} \qquad (30)$$

where $^B F_G$ is the gravitational force in the body frame and F_{G1}, F_{G2}, F_{G3} are the components of the gravity force in the x, y, and z direction, respectively.

b. Drag Force

The second force acting on the quadrotor is the drag force that opposes the movement of the quadrotor and is equal to the coefficient of drag multiplied by the square of the velocity in every axis. In the inertial frame the drag force is given by the equation:

$$^{u}F_{D} = \begin{bmatrix} \frac{1}{2}C_{Dx}A_{c}\rho\,^{u}V_{x}^{2} \\ \frac{1}{2}C_{Dy}A_{c}\rho\,^{u}V_{y}^{2} \\ \frac{1}{2}C_{Dz}A_{c}\rho\,^{u}V_{z}^{2} \end{bmatrix} \tag{31}$$

where A is the cross-sectional area, ρ is the air density, V_{x}, V_{y}, V_{z} are the body frame velocity components in the x, y, and z axis, respectively. C_{Dx}, C_{Dy}, C_{Dz} are the drag coefficients in each axis. In order to get the drag force components F_{Dx}, F_{Dy}, F_{Dz} in the x, y, and z direction in the body frame, we will also use the matrix $^{B}_{u}R$ from Eq. 9.

$$^{B}F_{D} = {}^{B}_{u}R\,^{u}F_{D} = \begin{bmatrix} \frac{1}{2}C_{Dx}A_{c}\rho\dot{x}^{2} \\ \frac{1}{2}C_{Dy}A_{c}\rho\dot{y}^{2} \\ \frac{1}{2}C_{Dz}A_{c}\rho\dot{z}^{2} \end{bmatrix} = \begin{bmatrix} F_{Dx} \\ F_{Dy} \\ F_{Dz} \end{bmatrix} \tag{32}$$

However, the velocity regime in which the quadrotor operates is at slow airspeeds. Avoiding the complex calculation of computational fluid dynamics and using the same considerations as in [1], where the coefficients of drag varies with the object's shape about ($C_{Dx} = C_{Dy} = C_{Dz} = 1.12$) and the cross-sectional area facing each axis and the combined surface area of the cross-frame with batteries and motors, are considered, then A_{c} is found to be about 0.16m for the x and y axis and 0.4m for the z-axis, respectively. Also, by using as the common air density value of $977.2\,Kg/m^{3}$, the resulting drag force components are found to be around $F_{Dx} = 8.756*10^{-5}*\dot{x}^{2}\,Kg/m, F_{Dy} = 8.756*10^{-5}*\dot{y}^{2}\,Kg/m, F_{Dz} = 5.289*10^{-4}*\dot{z}^{2}\,Kg/m$. This is quite negligible and thus can be ignored.

c. Thrust forces

The quadrotor's motion is only controlled by varying independently the speed of the four rotors. Each rotor produces both a thrust and a torque (moment) about its center of rotation. Let F_{T_i} be the thrust for its rotor, respectively, and l the distance of each rotor from the center of the quadrotor's mass. The quadrotor's motion is affected by the four propellers only in the z-direction so we have that:

$$F_T = \begin{bmatrix} 0 \\ 0 \\ F_{T1} + F_{T2} + F_{T3} + F_{T4} \end{bmatrix} = \begin{bmatrix} F_{Tx} \\ F_{Ty} \\ F_{Tz} \end{bmatrix} \tag{33}$$

where $F_{T1}, F_{T2}, F_{T3}, F_{T4}$ are the thrusts that the 1st, 2nd, 3rd and 4th rotor produces, respectively, and F_{Tx}, F_{Ty}, F_{Tz} are the total thrust force in each direction in body frame.

d. Total Force

The total force is the sum of the gravity, drag and thrust forces:

$$\sum {}^B F = \begin{bmatrix} F_{Dx} + F_{Tx} + F_{Gx} \\ F_{Dy} + F_{Ty} + F_{Gy} \\ F_{Dz} + F_{Tz} + F_{Gz} \end{bmatrix} =$$

$$= \begin{bmatrix} -mg\sin\theta \\ mg\sin\varphi\cos\theta \\ mg\cos\varphi\cos\theta \end{bmatrix} + \begin{bmatrix} \frac{1}{2}C_{Dx}A_c\rho\dot{x}^2 \\ \frac{1}{2}C_{Dy}A_c\rho\dot{y}^2 \\ \frac{1}{2}C_{Dz}A_c\rho\dot{z}^2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ F_{T1} + F_{T2} + F_{T3} + F_{T4} \end{bmatrix} \Rightarrow$$

$$\Rightarrow \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \begin{bmatrix} -mg\sin\theta + \frac{1}{2}C_{Dx}A_c\rho\dot{x}^2 \\ mg\sin\varphi\cos\theta + \frac{1}{2}C_{Dy}A_c\rho\dot{y}^2 \\ mg\cos\varphi\cos\theta + \frac{1}{2}C_{Dz}A_c\rho\dot{z}^2 + (F_{T1} + F_{T2} + F_{T3} + F_{T4}) \end{bmatrix} \tag{34}$$

8.   Moments (Torques) Calculation

Let τ_i be the torque generated by each propeller. The rolling moment $L(\tau_{roll})$ is produced by varying the right (#2) and the left (#4) rotor speeds

$$L = \tau_{roll} = \tau_2 - \tau_4 = F_{T_2}l - F_{T_4}l = (F_{T_2} - F_{T_4})l = (F_{right} - F_{left})l$$

Similarly the pitching moment $M(\tau_{pitch})$ (torque) is produced by varying the front (#1) and the back rotor (#3) speeds

$$M = \tau_{pitch} = \tau_1 - \tau_3 = F_{T_1}l - F_{T_3}l = (F_{T_1} - F_{T_3})l = (F_{front} - F_{back})l$$

Due to the third Newton's law, the drag of the propellers produces a yawing moment (torque) on the body of the quadrotor. Therefore, the total yawing moment is obtained from all the rotor speeds (clockwise and counterclockwise), is in opposite direction of the motion of the propellers and is given by

$$N = \tau_{yaw} = (F_{T_1} + F_{T_3} - F_{T_2} - F_{T_4})d = (F_{right} + F_{left} - F_{front} - F_{back})d$$

where d is the force to moment scaling factor calculated to be d=4N.m.

9.   Moments of Inertia Calculations

a.   *Moment of Inertia along x*

If the two motors (#1 and #3) along x axis and the main, central body of the quadrotor considered cylindrical in shape with mass m_1, m_3 *and* m_c, radius R_1, R_3 *and* R_c and height h_1, h_3 *and* h_c, respectively, as shown in Figure 34.
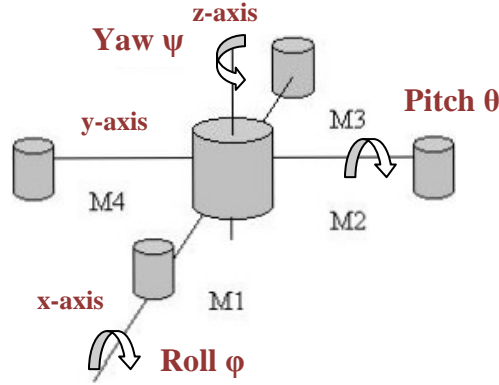
Figure 34.   Moments of inertia about x, y and z-axis.

Only the rolling motion is considered. So, the moment of inertia about the x axis would be due to the main body and the motors #1 and #3 motion about x-axis and due the motion of the other two motors (#2 and #4) about x axis

$$J_{xx} = J_c + J_1 + J_3 + J_{2,4} \tag{38}$$

The moment of inertia of a cylinder about an axis perpendicular to its body, as specified in Halliday- Resnick-Walker [57], thus the moment of inertia due to main body and the two motors are

$$J_{c,x} = \frac{m_c * r_c^2}{4} + \frac{m_c * h_c^2}{12} \tag{39}$$

$$J_{1,x} = \frac{m_1 * r_1^2}{4} + \frac{m_1 * h_1^2}{12} \tag{40}$$

$$J_{3,x} = \frac{m_3 * r_3^2}{4} + \frac{m_3 * h_3^2}{12} \tag{41}$$

The moment of inertia of two identical spheres connected together by a horizontal arm, and rotating about a vertical axis, which is passing through the center of the arm and is perpendicular to it, as also specified in [57], thus the moment of inertia due to motors #2 and #4 rotating about x-axis approximated to be

$$J_{2-4,x} = 2m_r l^2 \tag{42}$$

Since the motors are identical:

$$m_1 = m_3 = m_2 = m_4 = m_r \quad , \quad r_1 = r_3 = r_2 = r_4 = r_r \quad and \quad h_1 = h_3 = h_2 = h_4 = h_r$$

Finally, the total inertia about x axis by substituting Eq. 39 to 42 in Eq. 38 is

$$J_{xx} = \frac{mr_r^2}{2} + \frac{mh_r^2}{6} + \frac{m_o r_c^2}{4} + \frac{m_o h_c^2}{12} + 2m_r l^2 \tag{43}$$

b. Moment of Inertia along x

Similarly, because the quadrotor structure is symmetrical and the motors are identical the moment of inertia about y-axis would be exactly the same:

$$J_{yy} = \frac{mr_r^2}{2} + \frac{mh_r^2}{6} + \frac{m_o r_c^2}{4} + \frac{m_o h_c^2}{12} + 2m_r l^2 \tag{44}$$

c. Moment of Inertia about z-axis

The moment of inertia about z-axis, will be due to main body and due to all the motors. The moment of inertia due to the main body is found in [57] to be:

$$J_{c,z} = \frac{m_c r_c^2}{2} \tag{45}$$

The moment of inertia due to all the motors #1 to #4 is approximated to be:

$$J_{1-2-3-4,z} = 4m_r l^2 \tag{46}$$

Therefore, the total inertia about z-axis is:

$$J_{zz} = \frac{m_c r_c^2}{2} + 4m_r l^2 \tag{47}$$

All the above moments of inertia were calculated for the quadrotor and found to be as follows:

$$\begin{aligned} J_{xx} &= 0.03 Kgm^2 \\ J_{yy} &= 0.03 Kgm^2 \\ J_{zz} &= 0.04 Kgm^2 \end{aligned} \tag{48}$$

10. Final Equations of Motion

Combining the Eq.34 with the dynamic Eq. 21 we can get:

$$
\Rightarrow \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} -mg\sin\theta + \dfrac{1}{2}C_{Dx}A_c\rho\dot{x}^2 \\[2mm] mg\sin\varphi\cos\theta + \dfrac{1}{2}C_{Dy}A_c\rho\dot{y}^2 \\[2mm] mg\cos\varphi\cos\theta + \dfrac{1}{2}C_{Dz}A_c\rho\dot{z}^2 + (F_{T_1}+F_{T_2}+F_{T_3}+F_{T_4}) \end{bmatrix} - \begin{bmatrix} qw-rv \\ ru-pw \\ pv-qu \end{bmatrix} \Rightarrow
$$

$$
\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -g\sin\theta + \dfrac{1}{2m}C_{Dx}A_c\rho\dot{x}^2 + (rv-qw) \\[2mm] g\sin\varphi\cos\theta + \dfrac{1}{2m}C_{Dy}A_c\rho\dot{y}^2 + (pw-ru) \\[2mm] g\cos\varphi\cos\theta + \dfrac{1}{2m}C_{Dz}A_c\rho\dot{z}^2 + \dfrac{1}{m}(F_{T_1}+F_{T_2}+F_{T_3}+F_{T_4}) + qu-pv \end{bmatrix} \tag{49}
$$

If we neglect the drag force then the equation takes the form:

$$
\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} -g\sin\theta + (rv-qw) \\[2mm] g\sin\varphi\cos\theta + (pw-ru) \\[2mm] g\cos\varphi\cos\theta + \dfrac{1}{m}(F_{T_1}+F_{T_2}+F_{T_3}+F_{T_4}) + qu-pv \end{bmatrix} \tag{50}
$$

Similarly as before, if Eq. 17 is combined with Eq.34 and the transformation matrix in Eq. 9 is applied, then:

60

$$\sum{}^{B}F = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = {}_{u}^{B}Rm\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} \Rightarrow \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{1}{m}{}_{B}^{u}R\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} =$$

$$\frac{1}{m}{}_{B}^{u}R{}_{u}^{B}R\begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} + \frac{1}{m}{}_{B}^{u}R\begin{bmatrix} \frac{1}{2}C_{Dx}A_c\rho\dot{x}^2 \\ \frac{1}{2}C_{Dy}A_c\rho\dot{y}^2 \\ \frac{1}{2}C_{Dz}A_c\rho\dot{z}^2 \end{bmatrix}^{\approx 0} + \frac{1}{m}{}_{B}^{u}R\begin{bmatrix} 0 \\ 0 \\ F_{T_1}+F_{T_2}+F_{T_3}+F_{T_4} \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = {}_{B}^{u}R\frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ \frac{1}{m}(F_{T_1}+F_{T_2}+F_{T_3}+F_{T_4}) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \Rightarrow$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} \frac{1}{m}(F_{T_1}+F_{T_2}+F_{T_3}+F_{T_4})(\cos\phi\sin\psi+\cos\phi\sin\theta\cos\psi) \\ \frac{1}{m}(F_{T_1}+F_{T_2}+F_{T_3}+F_{T_4})(-\sin\phi\cos\psi+\cos\phi\sin\theta\sin\psi) \\ \frac{1}{m}(F_{T_1}+F_{T_2}+F_{T_3}+F_{T_4})\cos\phi\cos\theta - g \end{bmatrix} \tag{51}$$

Substituting Eqs. 35, 36 and 37 in Eq. 28 becomes:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \dfrac{\tau_2-\tau_4}{J_{xx}} - \dfrac{qr}{J_{xx}}(J_{zz}-J_{yy}) \\ \dfrac{\tau_1-\tau_3}{J_{yy}} - \dfrac{pr}{J_{yy}}(J_{xx}-J_{zz}) \\ \dfrac{(F_{T_1}+F_{T_3}-F_{T_2}-F_{T_4})d}{J_{zz}} - \dfrac{pq}{J_{zz}}(J_{yy}-J_{xx}) \end{bmatrix} = \begin{bmatrix} \dfrac{(F_{T_2}-F_{T_4})l}{J_{xx}} - \dfrac{qr}{J_{xx}}(J_{zz}-J_{yy}) \\ \dfrac{(F_{T_1}-F_{T_3})l}{J_{yy}} - \dfrac{pr}{J_{yy}}(J_{xx}-J_{zz}) \\ \dfrac{(F_{T_1}+F_{T_3}-F_{T_2}-F_{T_4})d}{J_{zz}} - \dfrac{pq}{J_{zz}}(J_{yy}-J_{xx}) \end{bmatrix}$$

(52)

Equations 15, 51 and 52 form the dynamical model of the quadrotor that will be used for the design of the controller. Although some approximations and assumptions already were made, some more simplifications have to be done for the better implementation of the control scheme.

61

B. MODELLING OF GROUND ROBOT QBOT

1. Differential Drive Kinematics

The notation and a mathematical model for the kinematics of a wheeled mobile robot will be developed in this section complying with references [56]-[61]. Wheeled robots have some simplifying features that make them easier to be modeled than real vehicles, since the operation field is on 2-D space environment. In particular, these robots have two independently driven, coaxial wheels. The speed difference between the two wheels causes a rotation of the robot about the center of the axis while the wheels produce motion in the forward or reverse direction. It is known that real vehicle dynamics, at high velocities, cause a vertical motion which is compensated for by applying suspension to their design. However, because these robots operate in very low speeds the vertical motion is almost negligible and no suspension is required.

Since the robot is moving on a horizontal plane, the degrees of freedom are three (3 DOF), the x and y positions along the axis of the differential drive and the rotation around z-axis:

$$P = \begin{bmatrix} {}^B x \\ {}^B y \\ {}^B \vartheta \end{bmatrix} \tag{53}$$

The following figure 34 depicts the kinematics of the iRobot Create, where the distance between the two wheels' centers is L=0.34m.
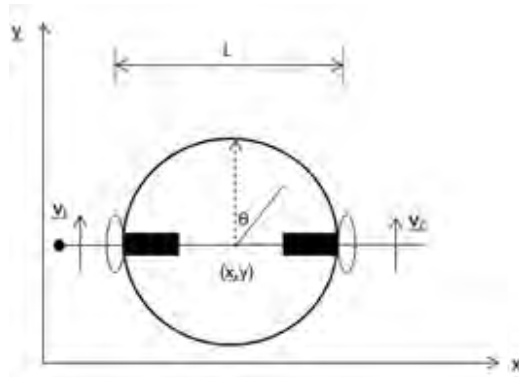
Figure 35.   Kinematics of the differential drive robot, iRobot Create.

62

As described before, in the case of the quadrotor, the motion has to be transformed from the body (robot here) frame to the inertial frame. The transformation matrix of Eq.9 will be "rotation around the z-axis" since we are dealing only with a 2-D frame:

$$_{B}^{u}R_{z} = \begin{bmatrix} \cos\vartheta & \sin\vartheta & 0 \\ -\sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So, Eq.53 becomes:

$$^{u}P = {}_{B}^{u}R_{z}\,{}^{B}P \Rightarrow \begin{bmatrix} {}^{u}x \\ {}^{u}y \\ {}^{u}\vartheta \end{bmatrix} = \begin{bmatrix} \cos\vartheta & \sin\vartheta & 0 \\ -\sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \vartheta \end{bmatrix} \Rightarrow$$

$$\Rightarrow \begin{bmatrix} {}^{u}x \\ {}^{u}y \\ {}^{u}\vartheta \end{bmatrix} = \begin{bmatrix} x\cos\vartheta + y\sin\vartheta \\ -x\sin\vartheta + y\cos\vartheta \\ \vartheta \end{bmatrix} \Rightarrow$$

$$\Rightarrow \begin{cases} {}^{u}x = x\cos\vartheta + y\sin\vartheta \\ {}^{u}y = -x\sin\vartheta + y\cos\vartheta \\ {}^{u}\vartheta = \vartheta \end{cases} \tag{54}$$

If the position of its wheels' centers are C_i , $i = 1,2$ for right and left wheel, respectively, the longitudinal component of velocity is $v_{long,i}$ and in lateral direction is $v_{lat,i}$, by differentiating Eq. 53 the velocity of C_i can be obtained in the body frame (2x1 vector):

$$^{B}v_{i} = \begin{bmatrix} {}^{B}v_{long,i} \\ {}^{B}v_{lat,i} \end{bmatrix} = \begin{bmatrix} {}^{B}\dot{x} \\ {}^{B}\dot{y} \end{bmatrix} \tag{55}$$

From body frame to the inertia frame:

$$^{u}\dot{x} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = {}^{u}_{B}R_{z}\,{}^{B}v = {}^{u}_{B}R\begin{bmatrix} u \\ v \end{bmatrix}$$

$$\begin{bmatrix} {}^{u}v_{long,i} \\ {}^{u}v_{lat,i} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\vartheta & \sin\vartheta & 0 \\ -\sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} {}^{B}v_{long,i} \\ {}^{B}v_{lat,i} \end{bmatrix} \Rightarrow$$

$$\Rightarrow \begin{bmatrix} {}^{u}v_{long,i} \\ {}^{u}v_{lat,i} \end{bmatrix} = \begin{bmatrix} {}^{B}v_{long,i}\cos\vartheta + {}^{B}v_{lat,i}\sin\vartheta \\ -{}^{B}v_{long,i}\sin\vartheta + {}^{B}v_{lat,i}\cos\vartheta \end{bmatrix} \Rightarrow$$

$$\Rightarrow \begin{bmatrix} {}^{u}v_{long,i} \\ {}^{u}v_{lat,i} \end{bmatrix} = \begin{bmatrix} \dot{x}\cos\vartheta + \dot{y}\sin\vartheta \\ -\dot{x}\sin\vartheta + \dot{y}\cos\vartheta \end{bmatrix} \tag{56}$$

Consider the rolling wheel with a point of contact P with the ground.
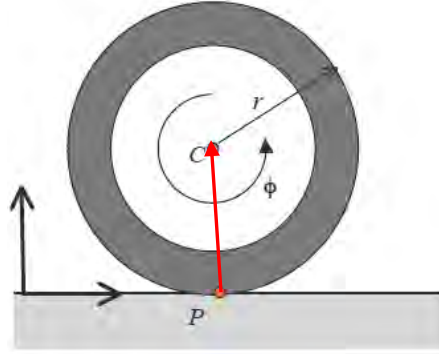
Figure 36.   Top View of the Robot

Because it is rolling, the velocity of point P is zero. Suppose the wheel rolls along the y axis on the y-z plane given by x = 0. If we recall from physics [57] that the velocities of two points A and B on a rigid body with angular velocity ω, are related by the equation:

$$v_{B} = v_{A} + \omega \times \overrightarrow{AB}$$

Then the velocity of the center C_i can be obtained from the previous equation:

$$v_{C,i} = v_P + \omega_i \times \overrightarrow{PC_i} \tag{57}$$

If $\dot{\varphi}_i, i = 1, 2$ each wheel's speed, respectively, then the velocity of the center C_i will be ($\hat{i}, \hat{j}, \hat{k}$ the unit vectors along x, y, z axis, respectively):

$$v_{C,i} = \omega_i \times r_i = \dot{\varphi}_i \hat{i} \quad \times \quad r\hat{k} = -r\dot{\varphi}_i \hat{j} \tag{58}$$

Since the robot cannot slide in a lateral direction, the velocity of the point C_i must be along the longitudinal, thus the velocity of the point C_i in the lateral direction must be zero:

$$v_{lat,i} = 0 \Rightarrow -\dot{x}_i \sin\vartheta + \dot{y}_i \cos\vartheta = 0$$

After rearranging terms:

$$\dot{x}_i \sin\vartheta = \dot{y}_i \cos\vartheta \Rightarrow \dot{x}_i \tan\vartheta = \dot{y}_i \tag{59}$$

Then the Eq. 65 will be equal to:

$$v_{long,i} = -r\dot{\varphi}_i \tag{60}$$

$$v_{lat,i} = 0$$

Next, φ_i is measured in a counterclockwise direction from one arbitrarily chosen side of the robot as shown in Figure 36.
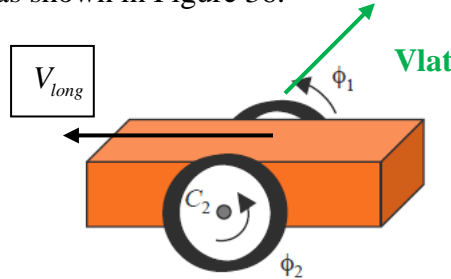
Figure 37.   The wheel angles (Positive in a counterclockwisedirection from one (arbitrarily chosen) side of the mobile robot). After [58].

Now consider the coordinates of the center of the axle (x, y) which is clearlyhalf way between C_1 and C_2 :

$$x = \frac{x_1 + x_2}{2}$$
$$y = \frac{y_1 + y_2}{2}$$

(61)

The velocity of the point C_i is given by:

$$v_{C,i} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \dfrac{\dot{x}_1 + \dot{x}_2}{2} \\ \dfrac{\dot{y}_1 + \dot{y}_2}{2} \end{bmatrix}$$

(62)

The longitudinal velocity component if we combine Eq.56, 60 and 62will become:

$$v_{long,i} = \dot{x}\cos\vartheta + \dot{y}\sin\vartheta = \frac{\dot{x}_1\cos\vartheta + \dot{y}_1\sin\vartheta}{2} + \frac{\dot{x}_2\cos\vartheta + \dot{y}_2\sin\vartheta}{2} = \frac{v_{C,1} + v_{C,2}}{2} \Rightarrow$$

$$\Rightarrow v_{long,i} = -\frac{r\dot{\varphi}_1 + r\dot{\varphi}_2}{2}$$

(63)

Now if we consider the two points C_1 *and* C_2 which are rigidly attached to theaxle and the robot, like Eq. 57, the velocities of these two points are related by theequation:

$$v_{C_2} = v_{C_1} + \dot{\vartheta}\hat{k} \times \overrightarrow{C_1 C_2}$$

(64)

That can be transformed to:

$$-r\dot{\varphi}_2 = -r\dot{\varphi}_1 + L\dot{\vartheta} \Rightarrow L\dot{\vartheta} = r\dot{\varphi}_1 - r\dot{\varphi}_2 \Rightarrow \dot{\vartheta} = \frac{r\dot{\varphi}_1 - r\dot{\varphi}_2}{L}$$

(65)

If the velocities of the left and right wheel are denoted as:

$$\dot{\varphi}_1 = \dot{\varphi}_{right} \quad and \quad \dot{\varphi}_2 = \dot{\varphi}_{left}$$

$$v_{C_1} = v_{right} \quad and \quad v_{C_2} = v_{left}$$

66

we have the two equations that relate the robot velocities tothe the wheels' speeds:

$$v_{long,i} = -\frac{r\dot{\varphi}_{right} + r\dot{\varphi}_{left}}{2} \tag{66}$$

$$\dot{\vartheta} = \frac{r\dot{\varphi}_{right} - r\dot{\varphi}_{left}}{L} \tag{67}$$

If we substitute Eq.59into Eq.57:

$$v_{long,i} = \dot{x}\cos\vartheta + \dot{x}\tan\vartheta\sin\vartheta = \dot{x}(\cos\vartheta + \frac{\sin^2\vartheta}{\cos\vartheta}) = \dot{x}(\frac{\cancel{\cos^2\vartheta + \sin^2\vartheta}}{\cos\vartheta})^{-1} \Rightarrow$$

$$\Rightarrow \dot{x} = v_{long,i}\cos\vartheta \tag{68}$$

$$Also \quad since \quad \dot{y} = \dot{x}\tan\vartheta = v_{long,i}\cancel{\cos\vartheta}\frac{\sin\vartheta}{\cancel{\cos\vartheta}} \Rightarrow$$
$$\Rightarrow \dot{y} = v_{long,i}\sin\vartheta \tag{69}$$

Finally, if we substitute the Eq.63 into the state equations of Eq.54, they can be written in the inertia frame as:

$$\dot{P} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\vartheta} \end{bmatrix} = \begin{bmatrix} \dfrac{v_{right} + v_{left}}{2}\cos\vartheta \\ \dfrac{v_{right} + v_{left}}{2}\sin\vartheta \\ \dfrac{(v_{right} - v_{left})}{L} \end{bmatrix} = \begin{bmatrix} -\dfrac{r(\dot{\varphi}_{right} + \dot{\varphi}_{left})}{2}\cos\vartheta \\ -\dfrac{r(\dot{\varphi}_{right} + \dot{\varphi}_{left})}{2}\sin\vartheta \\ \dfrac{r(\dot{\varphi}_{right} - \dot{\varphi}_{left})}{L} \end{bmatrix} = \begin{bmatrix} -\dfrac{r}{2}\cos\vartheta & -\dfrac{r}{2}\cos\vartheta \\ -\dfrac{r}{2}\sin\vartheta & -\dfrac{r}{2}\sin\vartheta \\ \dfrac{r}{L} & -\dfrac{r}{L} \end{bmatrix}\begin{bmatrix} \dot{\varphi}_{right} \\ \dot{\varphi}_{left} \end{bmatrix} \tag{70}$$

The velocities of the wheels are bounded such that:

$$-v_{\max} \le v_{left} \le v_{\max} \quad and \quad -v_{\max} \le v_{right} \le v_{\max} \tag{71}$$

where $v_{\max} = 0.5m/s$

The robot will move forward when $v_{right} > v_{left} > 0$, left when $v_{left} > v_{right} > 0$ and will spin in place when $v_{left} = v_{right}$, and not equal to zero.

Returning to the definition of a two wheel differential drive robot, it has two drive wheels mounted on a common axis and each wheel is controlled by a different motor. Therefore, each wheel is able to be driven at different velocities either forward or backward. As it is already reported before, by varying the velocity of each wheel, rolling motion can be achieved and the robot will rotate about a point that lies along the common left and right wheel axis. This point is widely known as the ICC - Instantaneous Center of Curvature [58] (See Figure 37).
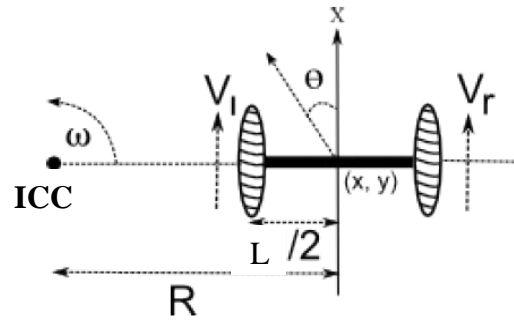
Figure 38.   Kinematics of differential robots. After [58].

The trajectory of the robot can be controlled by varying the velocities of the wheels. The rate of rotation ω about the ICC must be the same for both wheels. Hence, the following equations establish a relation between the motion parameters of a differential drive mobile robot.

$$v_{right} = \omega(R + \frac{L}{2}) \tag{72}$$

$$v_{left} = \omega(R - \frac{L}{2}) \tag{73}$$

where L is the distance between the centers of the two wheels, v_{right}, v_{left} are the right and left wheel velocities along the ground , and R is the signed distance from the ICC to the midpoint between the wheels.

The Eq.72 and 73 can be solved at any instance of time for R and ω as follows:

$$R = \frac{L(v_{right} + v_{left})}{2(v_{right} - v_{left})}$$ (74)

$$\omega = \frac{v_{right} - v_{left}}{L}$$ (75)

Figure 37 assumes that the robot is at some position (x, y) and is heading in a direction making an angle ϑ with the X axis. Knowing velocity v_{right}, v_{left}, and using Eq.74, 75, the ICC location ((ICC_x, ICC_y) is determined as:

$$ICC_x = x - R\sin\vartheta = x - \frac{L(v_{right} + v_{left})}{2(v_{right} - v_{left})}\sin\vartheta$$ (76)

$$ICC_y = y + R\cos\vartheta = y + \frac{L(v_{right} + v_{left})}{2(v_{right} - v_{left})}\cos\vartheta$$ (77)

2.   Forward Kinematics

The forward kinematics problem of a mobile robot for given wheel velocities and initial robot's configuration $(x, y, \vartheta)_{t=0}$ is the determination of the robot's position $(x, y, \vartheta)_t$ at any other time t [56].

a.   Robot's Position Relative to ICC Location

If non-varying velocities v_{right}, v_{left} are given, the ICC location (ICC_x, ICC_y) will be a fixed position. Hence, the robot's state at time $t + \Delta t$ can be expressed via its state at time t as follows:

$$\begin{bmatrix} x' \\ y' \\ \vartheta' \end{bmatrix} = \begin{bmatrix} \cos(\omega \times \delta t) & -\sin(\omega \times \delta t) & 0 \\ \sin(\omega \times \delta t) & \cos(\omega \times \delta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - ICC_x \\ y - ICC_y \\ \vartheta \end{bmatrix} + \begin{bmatrix} ICC_x \\ ICC_y \\ \omega \times \delta t \end{bmatrix}$$ (78)

where (x, y, ϑ) and (x', y', ϑ') are the robot's positions at time t and $t + \Delta t$, respectively. This equation describes the motion of a robot rotating about its ICC with a radius of curvature R and an angular velocity ω (see Figure 40).
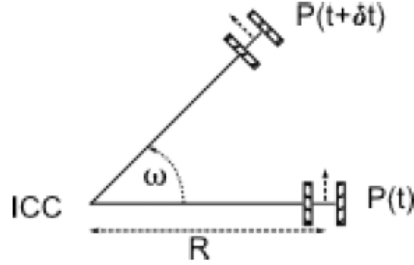
Figure 39.   Forward kinematics relative to ICC. After [58].

b.   *Robot's Position Relative to its Initial Position*

General motion equations of the robot capable of moving is in a particular direction $\vartheta(t)$ at a given velocity $v(t)$ are described as follows:

$$x(t) = \int_0^t v(t)\cos[\vartheta(t)]dt$$

$$y(t) = \int_0^t v(t)\sin[\vartheta(t)]dt \qquad (79)$$

$$\vartheta(t) = \int_0^t \omega(t)dt$$

For a differential drive robot, such as the Qbot, Eq. 79 becomes:

$$x(t) = \frac{1}{2}\int_0^t [v_{right}(t) + v_{left}(t)]\cos[\vartheta(t)]dt$$

$$y(t) = \frac{1}{2}\int_0^t [v_{right}(t) + v_{left}(t)]\sin[\vartheta(t)]dt \qquad (80)$$

$$\vartheta(t) = \frac{1}{L}\int_0^t [v_{right}(t) - v_{left}(t)]dt$$

If the Eq.79 is simplified, the robot's position at time $t + \Delta t$ will be:

70

$$
\begin{bmatrix} x' \\ y' \\ \vartheta' \end{bmatrix} = \begin{bmatrix} x + v\cos[\vartheta(t)]\delta t \\ y + v\sin[\vartheta(t)]\delta t \\ \vartheta + \omega\delta t \end{bmatrix}
\tag{81}
$$

Similarly, the Eq. 80 will become:

$$
\begin{bmatrix} x' \\ y' \\ \vartheta' \end{bmatrix} = \begin{bmatrix} x + \dfrac{1}{2}[v_{right}(t) + v_{left}(t)]\cos[\vartheta(t)]\delta t \\ y + \dfrac{1}{2}[v_{right}(t) + v_{left}(t)]\sin[\vartheta(t)]\delta t \\ \vartheta + \dfrac{1}{L}[v_{right}(t) - v_{left}(t)]\delta t \end{bmatrix}
\tag{82}
$$

For the special cases of $v_{right} = v_{left} = v$ and $-v_{right} = v_{left} = v$ in Eq.81, we

will have

$$
\begin{bmatrix} x' \\ y' \\ \vartheta' \end{bmatrix} = \begin{bmatrix} x + v\cos\vartheta\delta t \\ y + v\sin\vartheta\delta t \\ \vartheta \end{bmatrix}
\tag{83}
$$

$$
\begin{bmatrix} x' \\ y' \\ \vartheta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \vartheta + \dfrac{1}{L}2v\delta t \end{bmatrix} \quad,
\tag{84}
$$

respectively.

C.   CONTROL ARCHITECTURE OF THE TWO MODELS

Now that kinematics / dynamics for both vehicles have been developed, we can elaborate on Figures 30 and 31 and present overall feedback control architecture (Figure 41).
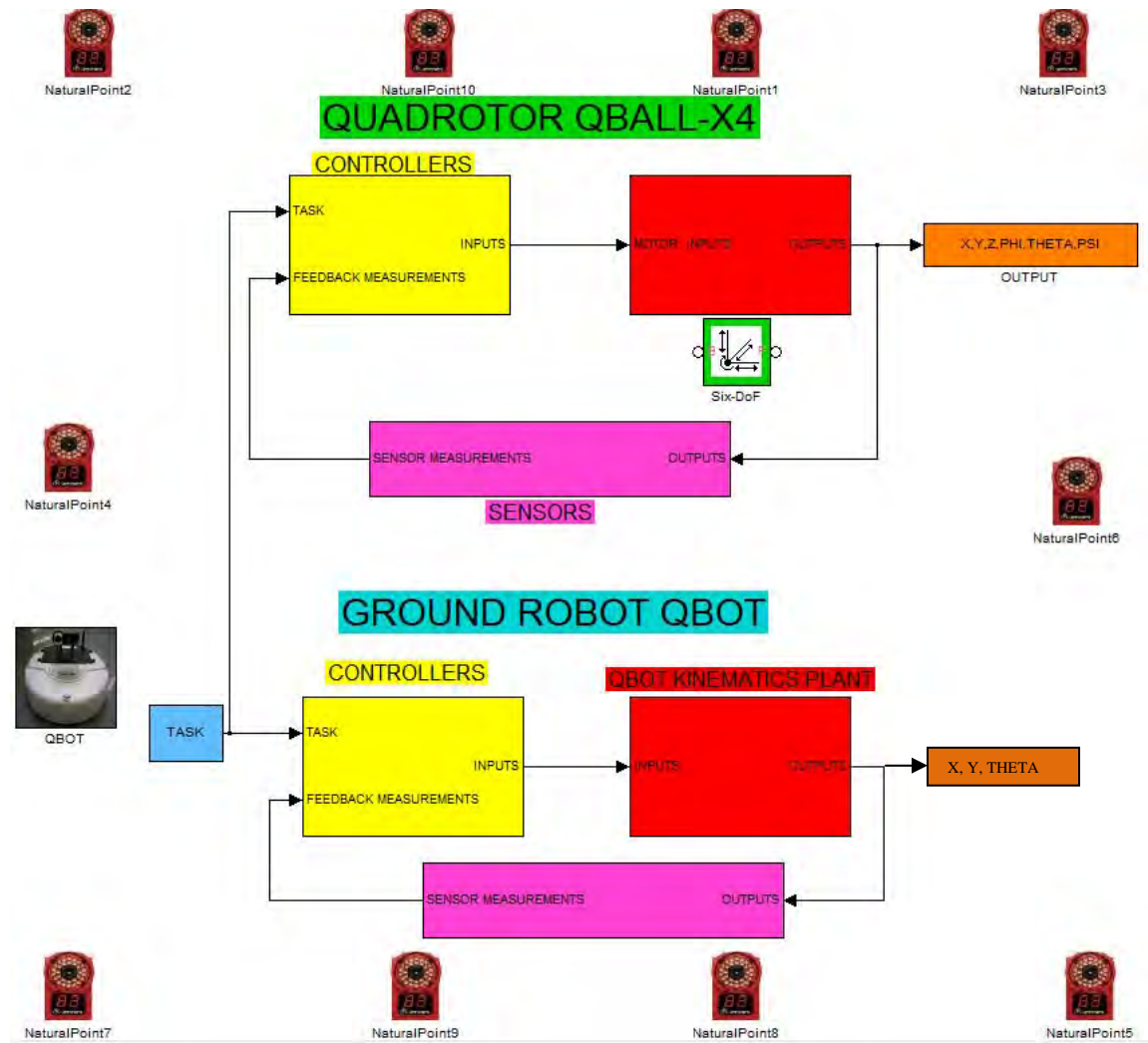
Figure 40.   Control Architecture

Figures 42 and 43 provide more details about the robot plant models and explain graphically the derivation of the equations of motion and how it will be implemented in the Simulink development environment.
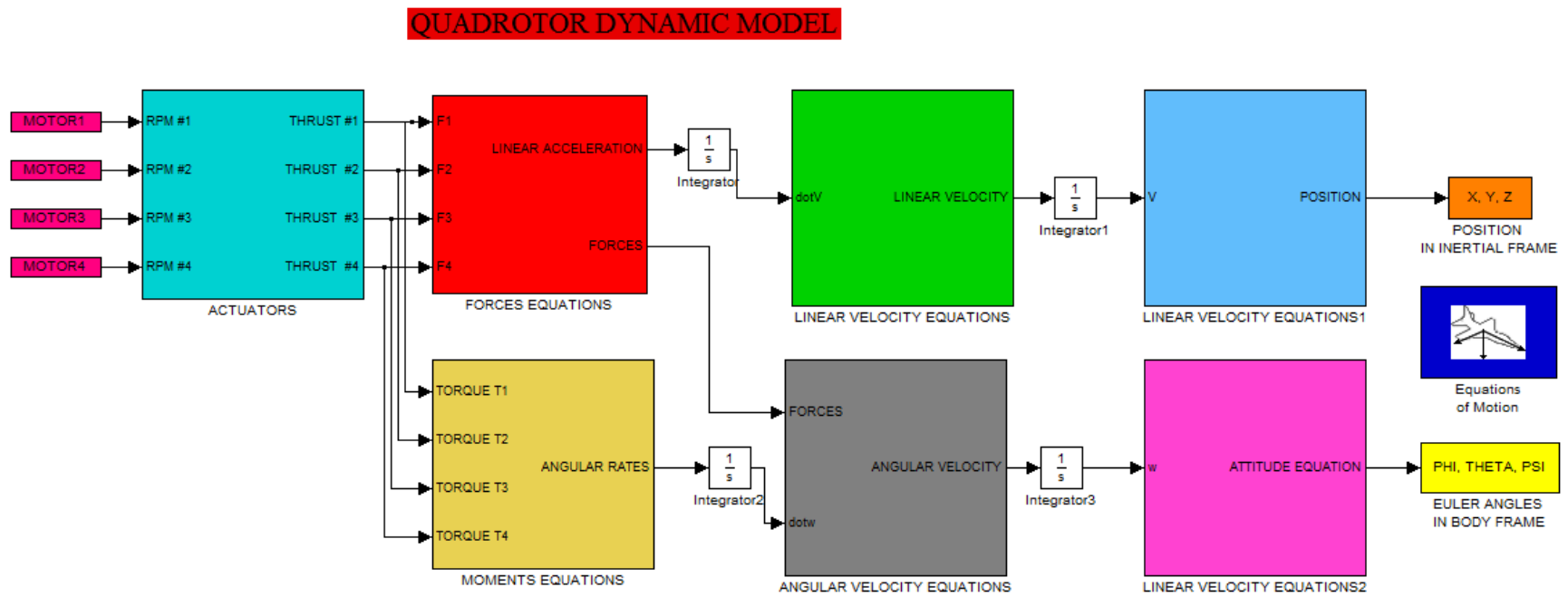
Figure 41.   Quadrotor QBall-X4 Dynamics Plant
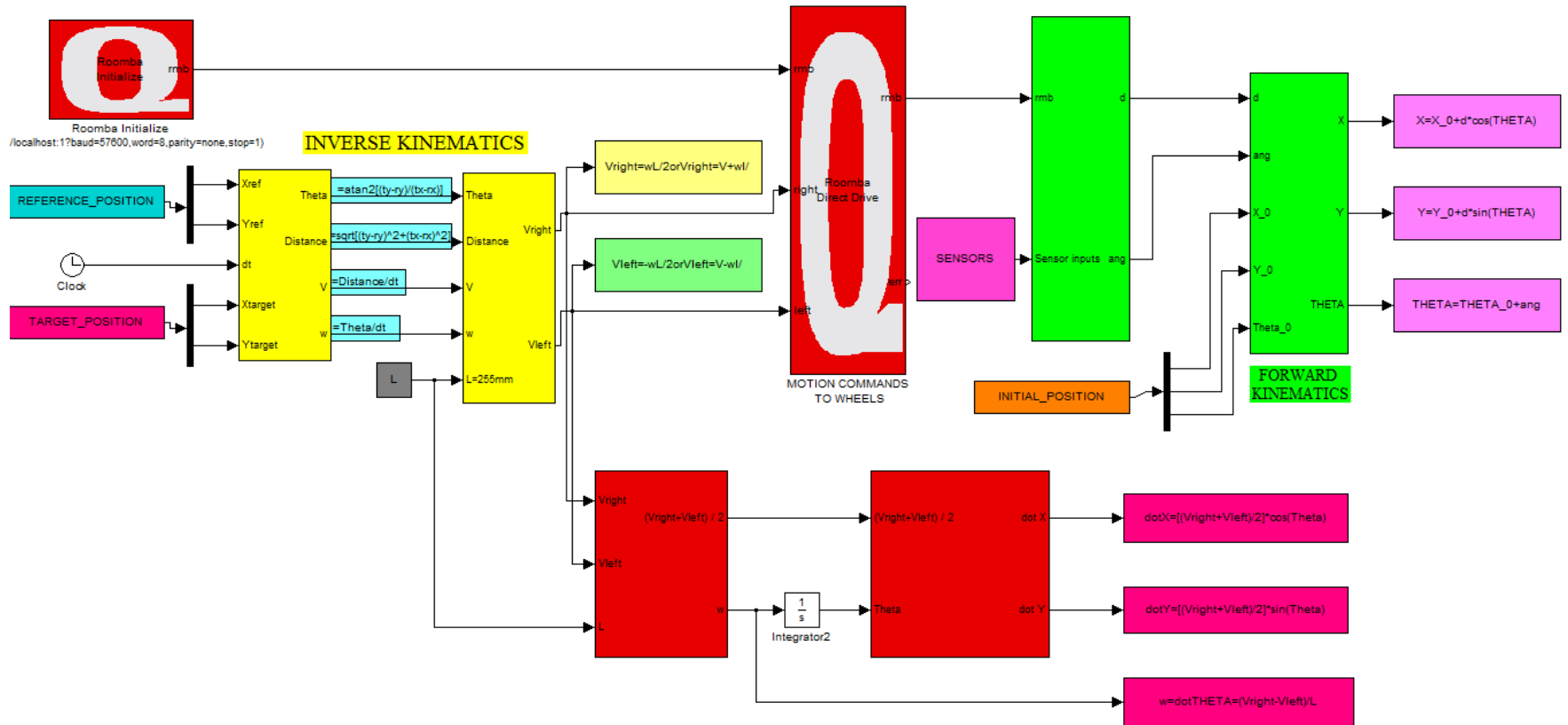
Figure 42. Ground Robot Qbot Kinematics Representation

IV. CONTROL STRATEGY

A. INTRODUCTION

The dynamic model of the quadrotor has been presented in many papers, so with some small variations, concerning the different assumptions to be done or the environment, it will be the same. However, the control schemes can be different, choosing from feedback linearization control, backstepping control, visual control and direct methods with inverse dynamics. In this chapter, the general control scheme will be presented. The first controller is the implemented one with the Linear Quadratic Regulator method imposed in five different channels, roll, pitch, x-y position, yaw and height. Although, the modeling took place for a nonlinear model, the nature of the experiments (laboratory – slow speeds) allows us to use a linearized model for the quadrotor control. Secondly, another controller is proposed using inverse dynamics in Virtual domain (IDVD) control technique for achieving a quasi-optimal solution in real time (tenths of second). This controller requires creating a trajectory generator with certain characteristics that will be described below and follows the quadrotor dynamics. Unfortunately, this controller was not implemented and tested for this model but it provides the steps and so far implementation for future work within the students already or starting working in this project.

B. IMPLEMENTED CONTROLLERS

1. Control Inputs

The quadrotor is controlled by independently varying the speed of the four rotors. For every attitude change the angular velocity of motors is changed, but the total thrust of all the four motors is constant in order to maintain the height. So, in order to produce a roll angle (ϕ) along the axis X_B, the angular velocity of the motor #2 is increased and the angular velocity of motor #4 must be decreased. Likewise, in order to produce a pitch angle (θ) along the axis Y_B, the angular velocity of the motors #3 must be increased, and the angular velocity of motor #1 must be decreased while at the same time the thrust

75

must be kept constant. It can be understood that yaw motion along the Z-axis of the body frame will be implemented by increasing total angular velocity of motors (1, 3) and decreasing the angular velocity of opposite rotation motors (2, 4). The rotors of quadrotor are located six inches from the end point of the aluminum rods and l is the length between the rotational axis of each rotor and the center of gravity of the quadrotor. So we can assign the following control inputs, U_i:

$$
\begin{aligned}
U_1 &= \frac{F_{T_1} + F_{T_3} + F_{T_2} + F_{T_4}}{m} \\
U_2 &= F_{T_2} - F_{T_4} \\
U_3 &= F_{T_1} - F_{T_3} \\
U_4 &= (F_{T_1} + F_{T_3} - F_{T_2} - F_{T_4})d
\end{aligned}
\tag{85}
$$

where F_{Ti} and d are, respectively, the normalized thrust force from each (ith) rotor and d is the force to moment scaling factor coefficient depending on the blade's Reynolds Number, Mach number and angle of attack.

2.   Waypoint Navigation

Waypoint navigation for the quadrotor will take place in two different ways. The Qball-X4 quadrotor will follow the position of the Qbot, which acts as the leader.

a.   Qbot's waypoint navigation via the Qbot model, an autonomous waypoint navigation program that moves Qbot through a series of predetermined waypoints chosen by the user.

b.   The quadrotor navigates by following the waypoints taken from the Qbot through the stream client of the real-time workshop. So, Qbot's Tx, Ty coordinates, are taken as inputs so a waypoint state machine can be built with the following states:

- Initialize
- Takeoff
- Follow waypoint
- Land

The waypoint state machine will allow quadrotor to follow the Qbot's position through its controllers. Alternatively, a trajectory generator with inverse dynamics can be used to generate new quasi-optimal trajectories for following the waypoints.

3.　　Linearization of Qball-X4 Control

Since the drag forces are negligible and the angles of the orientations are very small for hover flights [1], small angle approximation are invoiced:

:

$$\phi \ll 0.1 \Rightarrow \sin(\phi) \cong 0, \cos(\phi) \cong 1$$
$$\theta \ll 0.1 \Rightarrow \sin(\theta) \cong 0, \cos(\theta) \cong 1$$

(86)

So the kinematic Eq. 15, can be simplified to:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \cong \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -p \\ q \\ r \end{bmatrix}$$

(87)

If we take the derivatives of the previous equations and apply Eq.28 , then:

$$\ddot{\phi} = \dot{p} = \frac{U_2 l}{J_{xx}} - \frac{qr}{J_{xx}}(J_{zz} - J_{yy})$$

$$\ddot{\theta} = \dot{q} = \frac{U_3 l}{J_{yy}} - \frac{pr}{J_{yy}}(J_{xx} - J_{zz})$$

(88)

$$\ddot{\psi} = \dot{r} = \frac{U_4}{J_{zz}} - \frac{pq}{J_{zz}}(J_{yy} - J_{xx})$$

If Coriolis term is ignored, the simplified equations of motions can be derived as:

77

$$\ddot{\phi} = \dot{p} = \frac{U_2 l}{J_{xx}}$$

$$\ddot{\theta} = \dot{q} = \frac{U_3 l}{J_{yy}} \tag{89}$$

$$\ddot{\psi} = \dot{r} = \frac{U_4}{J_{zz}}$$

Eq. 56 can be written with the controls as follows:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} U_1(-\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi) \\ U_1(-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi) \\ U_1\cos\phi\cos\theta - g \end{bmatrix} \tag{90}$$

Hence, a simplified model can be presented with the state equation, that can be used for both controllers, the implemented and the proposed one:

$$\dot{X} = \frac{d}{dt} \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \phi \\ \theta \\ \psi \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ U_1(\cos\phi\sin\psi + \cos\phi\sin\theta\cos\psi) \\ U_1(-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi) \\ U_1\cos\phi\cos\theta - g \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dfrac{U_2 l}{J_{xx}} \\ \dfrac{U_3 l}{J_{yy}} \\ \dfrac{U_4 l}{J_{zz}} \end{bmatrix} = f(X,U) \tag{91}$$

where $X = [x, y, z, u, v, w, \phi, \theta, \psi, p, q, r]^T = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ (92)

and $\qquad\qquad\qquad\qquad U = [U_1, U_2, U_3, U_4]^T \tag{93}$

4. LQR Controllers / Channels Description

a. *Rotor (Actuator) Dynamics*

The thrust that is generated by each propeller is modeled by the first order

system equation:
$$F_{T\_i} = K \frac{\omega}{s+\omega} u_i \tag{93}$$

where u_i is the PWM input of the actuator, ω is the actuator bandwidth and K is a positive gain. Those parameters are already calculated through experimental studies in Quanser [51], and they are given in Table 7.

Parameter	Explanation	Value	Unit
K	PWM input of the actuator	120	N
ω	Actuator Bandwidth	15	rad/sec

Table 7. Thrust parameters

A state variable v_i will be used to represent actuator dynamics, which is defined as follows:

$$v_i = K \frac{\omega}{s+\omega} u_i \tag{94}$$

b. *Roll and Pitch Models*

Assuming that rotations about the x and y axes are decoupled, two propellers contribute to roll/pitch axis already modeled and discussed in the previous chapter. The rotation around the center of gravity is produced by the difference in the generated thrusts. If we set Δu the same for pitch and roll, the moments of inertia are found to be the same $J_{xx} = J_{yy} = J$ since the quadrotor already considered symmetric, the models can be formulated as also stated in [51]:

$$\ddot{\phi} = \dot{p} = \frac{U_2 l}{J_{xx}} = \frac{l}{J_{xx}} K \frac{\omega}{s+\omega} \Delta u$$

$$\ddot{\theta} = \dot{q} = \frac{U_3 l}{J_{yy}} = \frac{l}{J_{yy}} K \frac{\omega}{s+\omega} \Delta u$$

(95)

$$\Delta u = u_1 - u_3 \quad or \quad \Delta u = u_2 - u_4$$

If we combine these last three equations and the actuator dynamics the following state-space equations can be derived:

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \dfrac{Kl}{J} \\ 0 & 0 & -\omega \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \Delta u$$

(96)

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & \dfrac{Kl}{J} \\ 0 & 0 & -\omega \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \end{bmatrix} \Delta u$$

(97)

To facilitate the use of an integrator in the feedback structure a fourth state can be added to the state vector, which is defined as: $\dot{s} = \phi$ or $\dot{s} = \theta$ and augment this state to the state vector, the final roll and pitch models will be:

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{v} \\ \dot{s} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{Kl}{J} & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ v \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} \Delta u = A \begin{bmatrix} \phi \\ \dot{\phi} \\ v \\ s \end{bmatrix} + B \Delta u$$

(98)

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{v} \\ \dot{s} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{Kl}{J} & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ v \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} \Delta u = A \begin{bmatrix} \theta \\ \dot{\theta} \\ v \\ s \end{bmatrix} + B \Delta u$$

(99)

So the state space model has the same state and input matrices A and B and so the angles will be the same and only one computation should be done for the Gains in LQR problem. The LQR model for the pitch and roll controller is shown in Figure 44. The gains $K_{ph}i, K_{pi}i$ are the outputs of the Roll - Pitch LQR Controller m-file shown in the Appendix A.
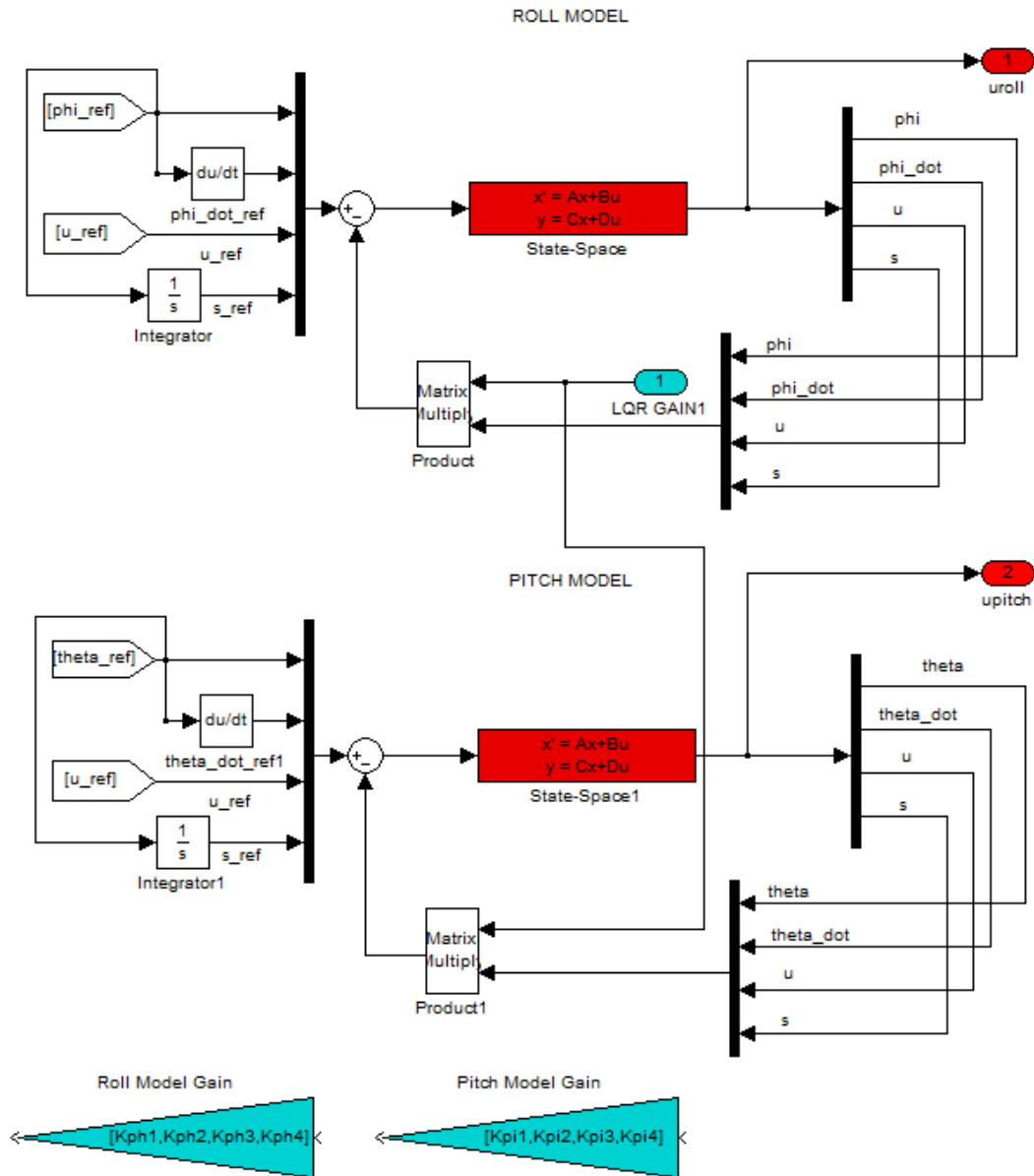
Figure 43.   Pitch – Roll LQR controller

c. Height Dynamics Model

The motion of the Qball-X4 in the vertical direction (along the Z axis) is affected by all the four propellers. Assuming that the same force thus PWM input is given for every propeller so the dynamic model of the Qball-X4 in this case can be written as:

$$\ddot{Z} = \frac{4F_T}{m}\cos\phi\cos\theta - g \qquad where \qquad F = Kv \qquad (100)$$

As expressed in this equation, if the roll and pitch angular rates are nonzero the overall thrust vector will not be perpendicular to the ground. Assuming that roll and pitch angles are close to zero, using again the Eq (94) for the state variable v and a fourth state for the use of an integrator in the feedback structure, the dynamic equations can be linearized to the following state space form:

$$\begin{bmatrix} \dot{Z} \\ \ddot{Z} \\ \dot{v} \\ \dot{s} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{4K}{m} & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Z \\ \dot{Z} \\ v \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ -g \\ 0 \\ 0 \end{bmatrix} \qquad (101)$$

The LQR model for the Height controller is shown in Figure 44. The gains $K_h i$ are the output of the Height LQR Controller m-file shown in the Appendix A.
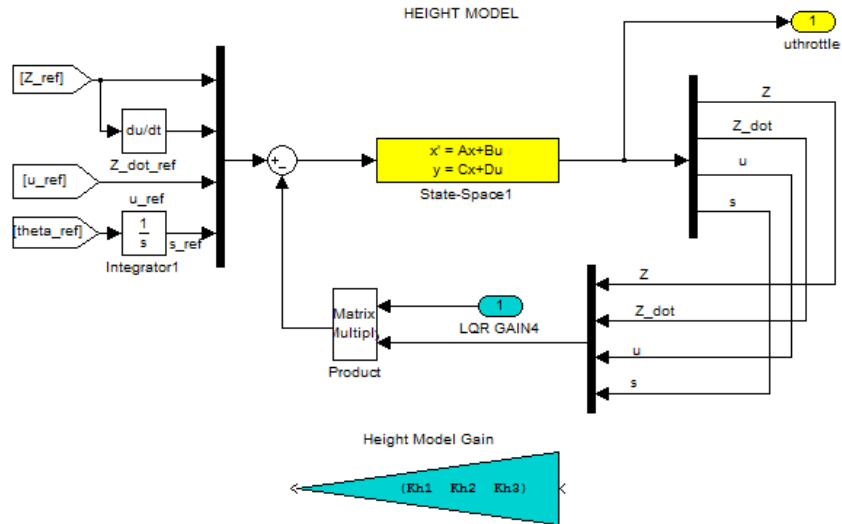
Figure 44.   Height LQR controller

d. X- Y Position Model

The motions of the Qball-X4 along the X and Y axes are caused by the total thrust and by changes of the roll/pitch angles. Assuming that the yaw angle is zero, the dynamics of motion in X and Y axes can be written as:

$$\ddot{x} = \frac{4F}{m}(\cos\phi\sin\psi + \cos\phi\sin\theta\cos\psi) \tag{102}$$

$$\ddot{y} = \frac{4F}{m}(-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi)$$

Assuming that the roll and pitch angle rates are close to zero, the following linear state-space equations can be derived for X and Y positions.

$$\ddot{x} = \frac{4K}{m}v\sin\theta \approx \frac{4K}{m}v\theta \text{ and}$$

$$\ddot{y} = \frac{4K}{m}v(-\sin\phi) = -\frac{4K}{m}v\phi$$

$$\begin{bmatrix} \dot{X} \\ \ddot{X} \\ \dot{v} \\ \dot{s} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{4K}{m}\theta & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ \dot{X} \\ v \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} u \tag{103}$$

$$\begin{bmatrix} \dot{Y} \\ \ddot{Y} \\ \dot{v} \\ \dot{s} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{4K}{m}\phi & 0 \\ 0 & 0 & -\omega & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} Y \\ \dot{Y} \\ v \\ s \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \omega \\ 0 \end{bmatrix} u \tag{104}$$

The LQR model for the X-Y Position controller is shown in Figure 46. The gains $K_x i, K_y i$ are the outputs of the X-Y Position LQR Controller m-file shown in the Appendix A.
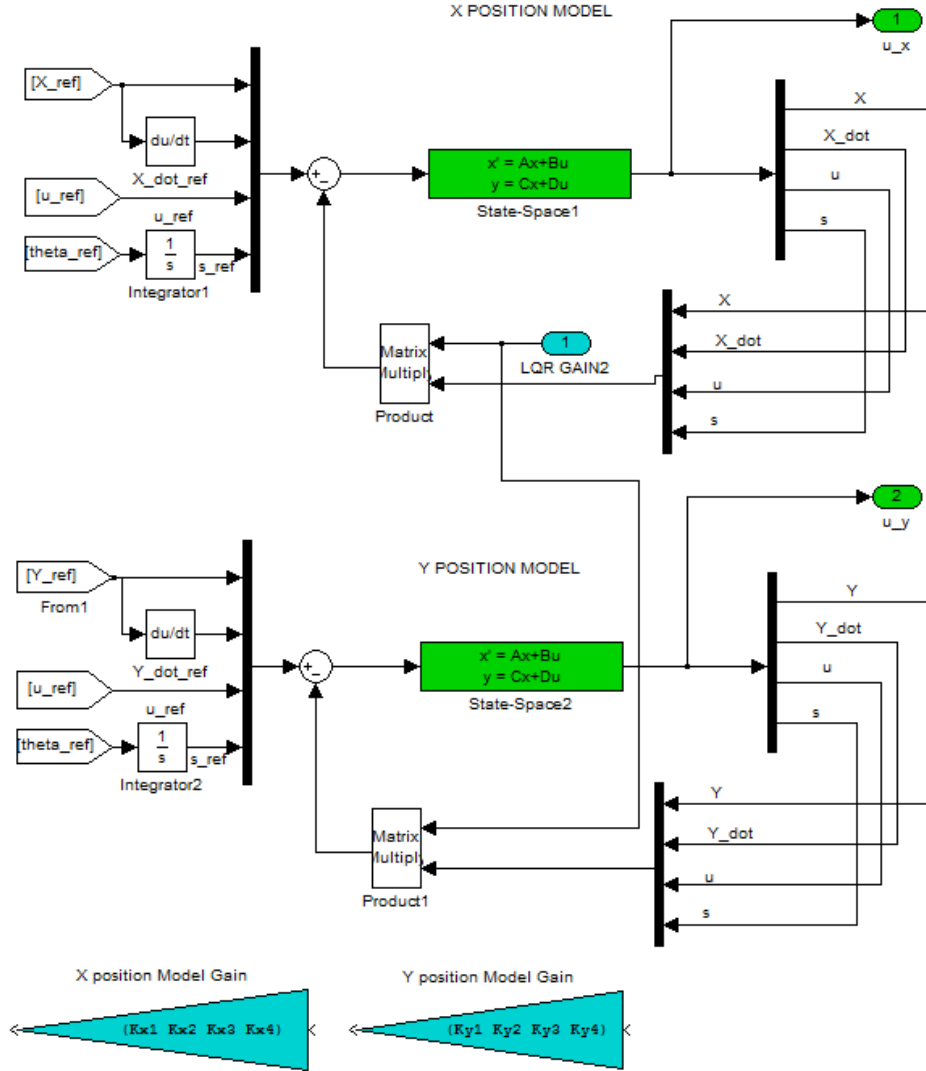
Figure 45.   X-Y Position LQR controller

e.   *Yaw Model*

As derived in the previous chapter the motion in the yaw axis is caused by the difference between the torques exerted by the two clockwise and the two counterclockwise rotating props. The motion in the yaw axis can be modeled by:

$$\ddot{\psi} = \dot{r} = \frac{U_4}{J_{zz}} = \frac{K_{yaw}}{J_{zz}} \Delta u$$

$$\Delta u = u_1 + u_3 - u_2 - u_4$$

(105)

The yaw axis dynamics can be rewritten in the state-space form as:

$$\begin{bmatrix} \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{K_{yaw}}{J_{zz}} \end{bmatrix} \Delta u \tag{106}$$

The LQR model for the yaw controller is shown in Figure 47. The gains $K_{yaw}i$ are the output of the Yaw LQR Controller m-file shown in the Appendix A.
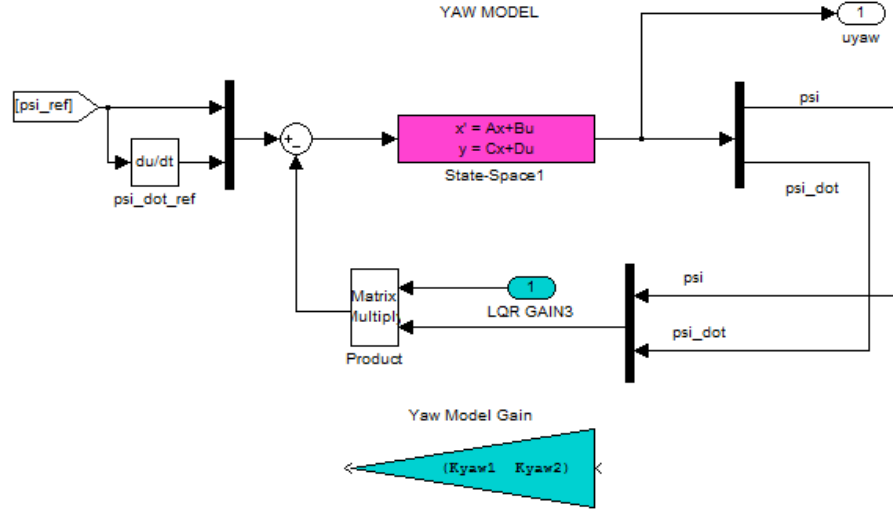
Figure 46.   Yaw LQR controller

f.   *Motor Inputs*

The main goal of the controller is to accomplish an algorithm of quadrotor control and provides decoupling of control channels in steady state. Thus the control inputs from the controller about each axis v_ϕ, v_θ, v_ψ , together with the throttle command for each rotor are combined to generate the control inputs u_1, u_2, u_3, u_4 that are given as follows:

$$u_1 = u_{th} + v_\theta + v_\psi$$
$$u_2 = u_{th} + v_\phi - v_\psi$$
$$u_3 = u_{th} - v_\theta + v_\psi$$
$$u_4 = u_{th} - v_\phi - v_\psi$$

This control mixing that take place in Figure 48 can be proved to be valid within the following derivations, showing that every control achieves the desired movement in each associated axis:

$$U_1 = u_1 + u_2 + u_3 + u_4 = u_{th} + v_\theta + v_\psi + u_{th} + v_\phi - v_\psi + u_{th} - v_\theta + v_\psi + u_{th} - v_\phi - v_\psi \Rightarrow$$
$$\Rightarrow U_1 = 4u_{th}$$

$$U_2 = u_1 - u_3 = u_{th} + v_\theta + v_\psi - (u_{th} - v_\theta + v_\psi) \Rightarrow$$
$$\Rightarrow U_2 = 2v_\theta$$

$$U_3 = u_2 - u_4 = u_{th} + v_\phi - v_\psi - (u_{th} - v_\phi - v_\psi) \Rightarrow$$
$$\Rightarrow U_3 = 2v_\phi$$

$$U_4 = u_1 + u_3 - u_2 - u_4 = u_{th} + v_\theta + v_\psi + u_{th} - v_\theta + v_\psi - (u_{th} + v_\phi - v_\psi + u_{th} - v_\phi - v_\psi) \Rightarrow$$
$$\Rightarrow U_4 = 2v_\psi$$
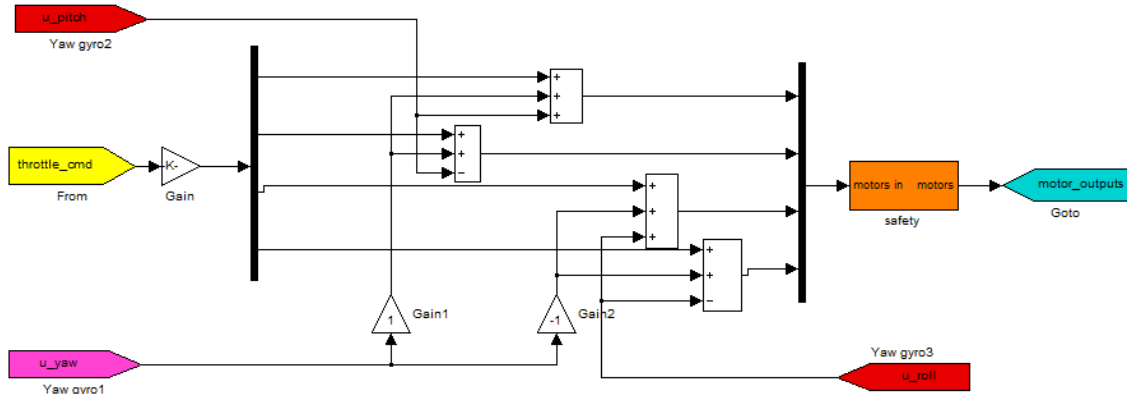
Figure 47.   Control Mixing Block

Figure 48.

5.   Qbot Inverse Kinematics Procedure

The model of the Qbot described in the previous chapter was described with the forward kinematics. The controller that the robot will use is introducing the inverse kinematics problem. The inverse kinematics problem of a mobile robot with initial configuration $(x, y, \vartheta)_{t=0}$ is the achievement of robot's target configuration $(x, y, \vartheta)_t$. If $(x, y, \vartheta)_{t=0} = (0, 0, 0)$, then solving the Eq.89, we will have:

$$x(t) = \frac{L[v_{right}(t) + v_{left}(t)]}{2(v_{right}(t) - v_{left}(t))} \sin[\frac{t}{L}(v_{right}(t) - v_{left}(t))]$$

$$y(t) = \frac{L[v_{right}(t) + v_{left}(t)]}{2(v_{right}(t) - v_{left}(t))} \cos[\frac{t}{L}(v_{right}(t) - v_{left}(t))] + \frac{L[v_{right}(t) + v_{left}(t)]}{2(v_{right}(t) - v_{left}(t))} \quad (107)$$

$$\vartheta(t) = \frac{L[v_{right}(t) + v_{left}(t)]}{2(v_{right}(t) - v_{left}(t))}$$

For a known time t and a target position (x, y), the last Eq. 107 will be solved for v_{right}, v_{left}. Eqs. 83 and 84 provide a simpler strategy to drive a robot to a target position $(x, y, \vartheta)_t$ where the robot can be rotated in place until it will be aimed toward (x, y), then driven forward until it will be at (x, y), and finally rotated in place until the required target orientation ϑ is met. From the first two equations of Eq. 80 we obtain

$$V_{right} + V_{left} = 2\sqrt{\dot{x}^2 + \dot{y}^2} \quad (108)$$

Also, form the same two equations we obtain $\vartheta = arc\tan\left(\frac{\dot{y}}{\dot{x}}\right)$ \quad (109)

Differentiating Eq.110 yields $\dot{\vartheta} = \frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2}$ \quad (110)

From the other hand, the third equation of Eq.80 reads $\dot{\vartheta} = \frac{V_{right} - V_{left}}{L}$ \quad (111)

Hence, $V_{right} - V_{left} = L\frac{\ddot{y}\dot{x} - \ddot{x}\dot{y}}{\dot{x}^2 + \dot{y}^2}\dot{\vartheta}$ \quad (112)

Eq.108 and 112 resolved for Vright and Vleft yield the controls required to follow a predetermined trajectory y(x).

C.   DIRECT METHOD BASED CONTROLLERS

1.   Introduction

The classical indirect methods cannot handle complicated problems in real time. There is a need for different techniques that simplify the problem or use numerical algorithms to provide near-optimal rather than optimal solutions in real time. The proposed controller introduces one of the direct methods already worked in real time. It is quite easy to program and provides effective optimization with feasible solutions. Taking Prof. Taranenko's ideas in early 60s as motivation, the computer's rapid development helped several engineers develop algorithms for real-time on-board calculation of quasi-optimal trajectories ([41]-[43]) for combat vehicles and missiles. There were successful tests within the Pilot's Associate program onboard 5th-generation aircraft [44]. This direct method can be used for unmanned air vehicles (UAVs) to generate approach trajectories.

The direct method's simplicity gives one the opportunity to develop the theory, and eventually test a real-time trajectory optimization scheme for the quadrotor. In the following sections the general architecture of the proposed controller will be developed, consisting of the trajectory generator and the trajectory follower. Using the parameters and dynamics of the already working controller we will introduce the trajectory generator which will generate optimal or quasi optimal feasible trajectories so that follow the waypoint pattern of the ground robot. This real time capability provides regeneration of each trajectory during the mission through feedback from the sensors. It allows for changing the objectives for any disturbances. To accomplish this task, a reference trajectory and the controls have to be found, so that even with LQR controller, the following of the path will be also accomplished. Also, an interpolator must be used to provide samples of the reference trajectory at the desired (high frequency) rate. The proposed scheme is shown in Figure 49. This control scheme differs from the previous one in the fact that in order to achieve a solution, the trajectory generator and the controller both need to follow the reference trajectory.
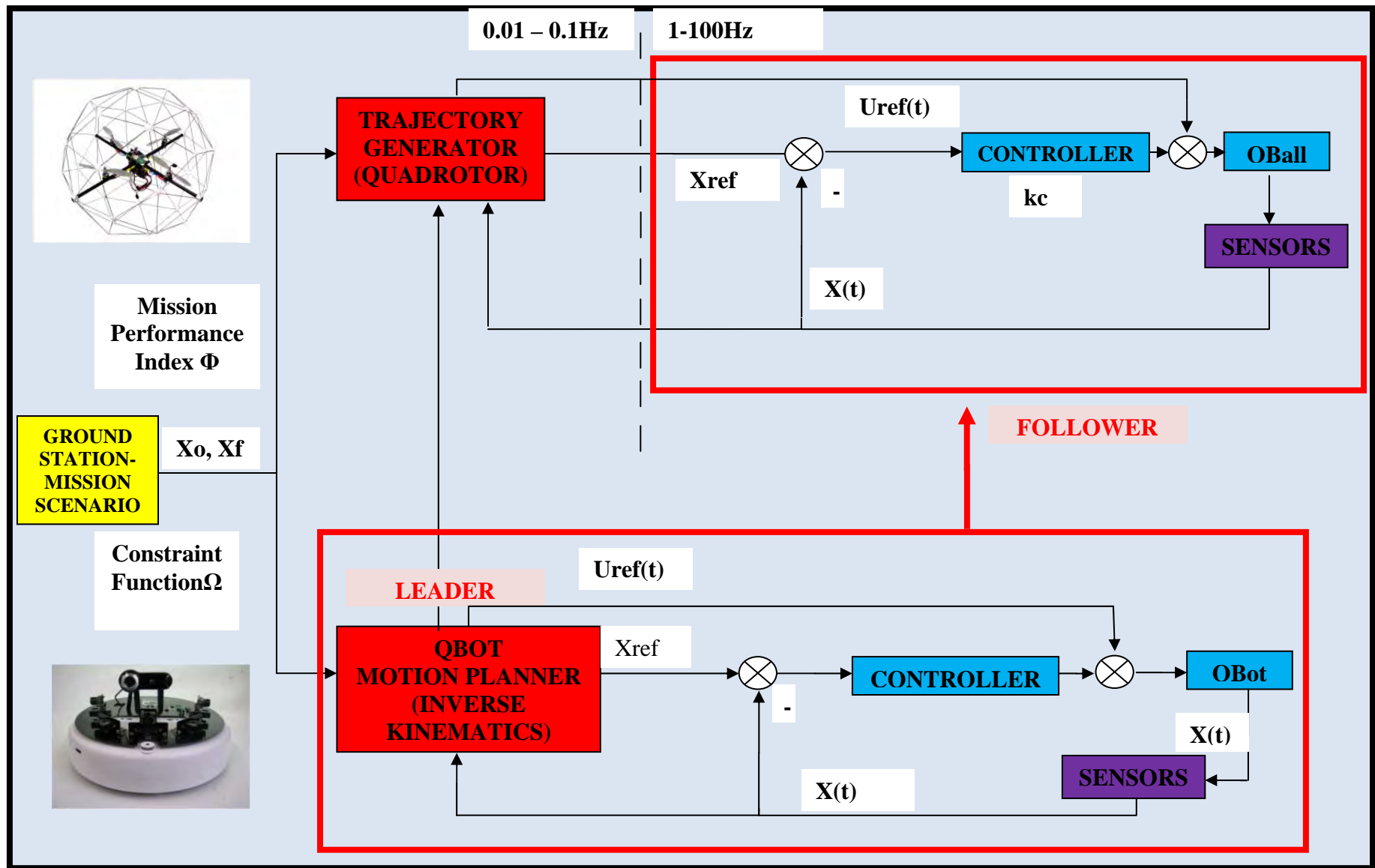
Figure 49. General Architecture of Joint Quadrotor – Robot Controller

2.   Formulation of the LQR Problem

The objective is to find a controller that provides the best possible performance with respect to the given criterion. The requirements of the controller are:

- Optimality
- Stability of the closed loop system
- Desirable gain and phase margin
- Robustness with respect to unmodelled dynamics of the plant and environmental uncertainties

For a linear state-space model of the system dynamics:

$$\dot{X}(t) = AX(t) + BU(t) \tag{113}$$

With an assumption of availability of measurements of full state the matrix gain K_c of the optimal control input must be determined:

$$U(t) = -K_c X(t) \tag{114}$$

So as to minimize the performance criterion or the cost function

$$J = \int_0^\infty (X^T Q X + U^T R U) dt \tag{115}$$

where Q and R, are positive definite Hermitean or real symmetric weight matrices. In fact Q must be positive semi definite.

Bryson's Rule defines an initial choice of matrices Q and R

$$Q_{ii} = \frac{1}{\max acceptable\_value\_of\_X_i^2}, i \in \{1, 2..., n\}$$

$$R_{jj} = \frac{1}{\max acceptable\_value\_of\_U_j^2}, j \in \{1, 2..., n\} \tag{116}$$

which corresponds to the following criteria

$$J_{LQR} = \int_0^\infty (\sum_1^n Q_{ii} X_i^2(t) + \sum_1^n R_{jj} U_j^2(t)) dt \tag{117}$$

90

For a time dependent reference trajectory $X_{ref}(t)$, the LQR control can be applied as a trajectory follower to minimize small errors between the measured state x and the reference state X_{ref}, and the control algorithm will become:

$$U(t) = U_{ref}(t) - K_c(X(t) - X_{ref}(t)) \qquad (118)$$

We have to compute the gain matrix K_c using a linearized quadrotor's model and then apply this control the non-linear model.

3.   Stability Analysis

Since the linearization of the quadrotor plant is already taken place at hover condition, the control gains K_c were designed with the weighting matrices as follows:

$$Q = 10^{-5} I_{12x12} \text{ and } R = diag(10^{-5}, 10^8, 10^8, 10^8),$$

if it is sure that the actuator constraints are maintained as stated in [62]. However, following the generated trajectory the hover condition will be sometimes violated. That's why a specific envelope where the operation will be stable has to be determined. As already derived in [63] a linearized stability set representing a circumference of a circle with a radius of 48 degrees can be practical enough:

$$s(t) = \left\{ \theta, \phi : \theta^2 + \phi^2 \le r^2, r = 48° \right\}$$

So if an extra constraint added to the optimization, that maintains angles ϕ and θ within this stability set within the trajectory generator the linearized time-invariant stability will be assured.

Now that we set up the problem, we can see the whole flow procedure within a diagram, in order to visualize better the technique to be used. The direct method optimization flow procedure is presented in the following chart, where each component shown will be described in the following sections of the chapter:
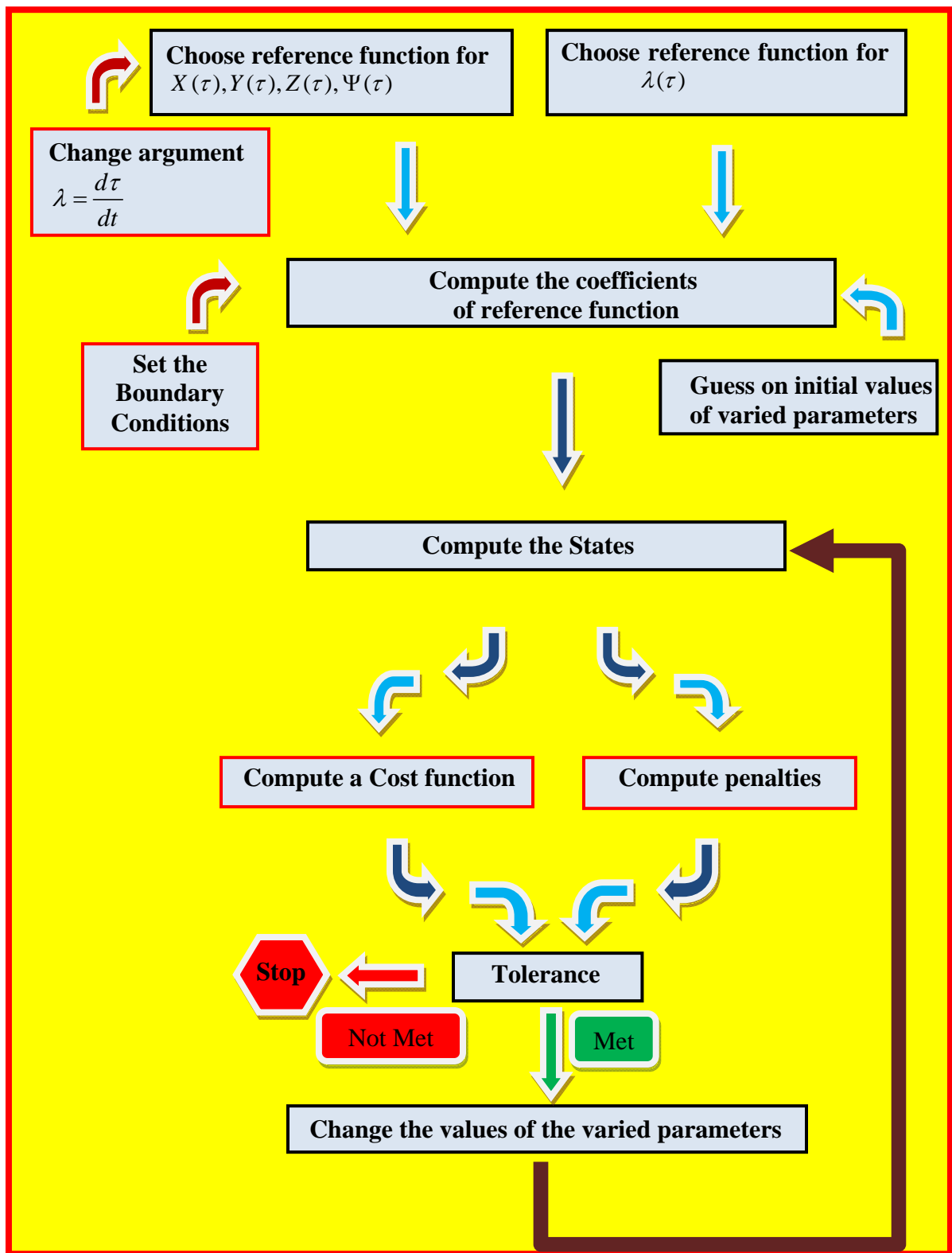
Figure 50.   Direct Method Optimization Flow Procedure

4.   Reference Trajectory

To approximate the Cartesian coordinates of a vehicle, its speed and its acceleration (six states), we can use functions that will define the variety of accessible trajectories and their choice will depend on the particular problem. In general, the more terms those functions have, the more accurate (closer to the really optimal) solution can be found. Once the Cartesian coordinates, speed and acceleration are defined using reference functions, the remaining states and controls are determined using inverse dynamics of original non-linear equations driving the system's dynamics. By using the direct method as Taranenko suggested in [63], a big advantage over the indirect method sis obtained since we eliminate the issue of solving the Cauchy problem for determining the trajectory with the given initial states and control time-histories, but instead, introducing the desired trajectory from the beginning, time-histories for all the controls are retrieved through the use of the inverse dynamics.

One modification of Taranenko's method is to employ elementary polynomials [44], [45], [62]-[64] as the reference functions. There are other alternatives, like Chebyshev polynomials [65], [66] or Laguerre polynomials [67] and others. In order to compute the coefficients, let us consider as reference function, algebraic polynomials of degree "n" for x, y, z coordinates and use as argument the virtual arc "τ", given as follows (the exact same procedure will take place for the other coordinates, y and z):

$$
\begin{aligned}
x_i(\tau) &= \sum_{k=0}^{n} a_{ik} \frac{(\max(1, k-2))! \tau^k}{k!} \\
x_i^{'}(\tau) &= \sum_{k=1}^{n} a_{ik} \frac{(\max(1, k-2))! \tau^{k-1}}{(k-1)!} \\
x_i^{''}(\tau) &= \sum_{k=2}^{n} a_{ik} \tau^{k-2} \\
x_i^{'''}(\tau) &= \sum_{k=3}^{n} (k-2) a_{ik} \tau^{k-3}
\end{aligned}
\tag{119}
$$

The degree "n" of these polynomials is chosen from the boundary conditions, where they have to be specified accordingly so that all the coefficients a_{ik} will be determined algebraically. The higher the maximum degree of the time derivative of a

93

vehicle coordinate at initial and end points, the higher the degree of the polynomial. The minimum degree of the polynomial to be chosen will be given by the equation:

$$n = d_0 + d_f + 1 \tag{120}$$

where d_0, d_f are the maximum orders of the time derivative of the coordinates at the initial and end points, respectively. So that at the boundary values for the quadrotor coordinates, the first and second time derivatives at both ends of the trajectory are satisfied, fifth-order polynomials should be chosen since $d_0 = d_f = 2$. Applying Eq. (85) we define eighteen unknown coefficients.

Generally, the final part of the trajectories needs to be smoother, since the control has to be more accurate while at landing or at rendezvous point. That's why $d_f = 3, x'''_{if} = 0 (i = 1, 2, 3)$ is usually proposed.

In case of $d_0 = 2, d_f = 3$, thus n = 6, where an additional optimization parameter is applied, then 24 coefficients a_{ik} are obtained. Subsequently, if we add two parameters, the coefficients will be 36.

Expressing the six coefficients $a_{Xk}, k = 0, 1, ... 5$ (the same manner for a_{Yk} and a_{Zk}) as a linear matrix equation, we will have [64]:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{12}\tau_f^4 & \frac{1}{20}\tau_f^5 \\ 0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{3}\tau_f^3 & \frac{1}{4}\tau_f^4 \\ 0 & 0 & 1 & \tau_f & \tau_f^2 & \tau_f^3 \end{bmatrix} \begin{bmatrix} a_{X0} \\ a_{X1} \\ a_{X2} \\ a_{X3} \\ a_{X4} \\ a_{X5} \end{bmatrix} = \begin{bmatrix} X_0 \\ \dot{X}_0 \\ \ddot{X}_0 \\ X_f \\ \dot{X}_f \\ \ddot{X}_f \end{bmatrix} \tag{121}$$

94

The reference functions provide us with the flexibility to increase the order of approximation and derivatives at both ends using them as additional varied parameters. In the previous case, the only varied parameter is τ_f (since all coefficients are determined from the boundary conditions), shown in Figure 54. The diagrams are produced very easily if you calculate the equation 121 in Matlab.
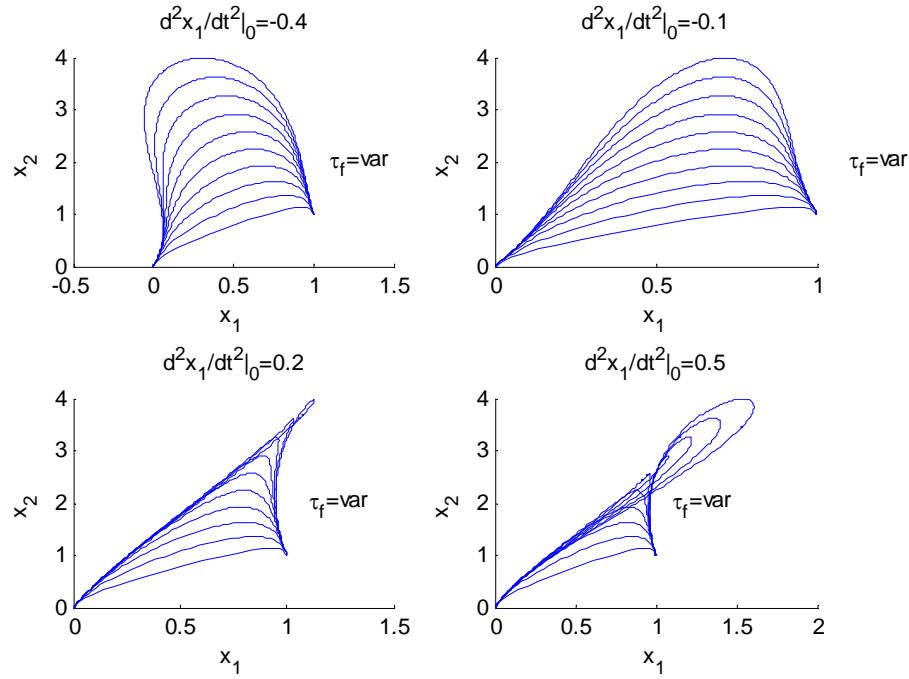
Figure 51.   Variation of the parameter of the reference functions, $\tau_f = \text{var}$

If 7th-order polynomials are used, the initial and final state, first and second derivatives were used as constraints attempted to be satisfied and the third derivatives at both ends of the trajectory became free variables along with the virtual arc length [64]:

$$x_i^{'''}(\tau) = P_{x'''i}(\tau) = a_{i3} + a_{i4}\tau + a_{i5}\tau^2 + a_{i6}\tau^3 + a_{i7}\tau^4$$

$$x_i^{''}(\tau) = P_{x''i}(\tau) = a_{i2} + a_{i3}\tau + \frac{1}{2}a_{i4}\tau^2 + \frac{1}{3}a_{i5}\tau^3 + \frac{1}{4}a_{i6}\tau^4 + \frac{1}{5}a_{i7}\tau^5 \qquad (122)$$

$$x_i^{'}(\tau) = P_{x'i}(\tau) = a_{i1} + a_{i2}\tau + \frac{1}{2}a_{i3}\tau^2 + \frac{1}{6}a_{i4}\tau^3 + \frac{1}{12}a_{i5}\tau^4 + \frac{1}{20}a_{i6}\tau^5 + \frac{1}{30}a_{i7}\tau^6$$

$$x_i(\tau) = P_{xi}(\tau) = a_{i0} + a_{i1}\tau + \frac{1}{2}a_{i2}\tau^2 + \frac{1}{6}a_{i3}\tau^3 + \frac{1}{24}a_{i4}\tau^4 + \frac{1}{60}a_{i5}\tau^5 + \frac{1}{120}a_{i6}\tau^6 + \frac{1}{210}a_{i7}\tau^7$$

95

The coefficients will be computed by solving the following system of linear algebraic equations [64]:

$$
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{24}\tau_f^4 & \frac{1}{60}\tau_f^5 & \frac{1}{120}\tau_f^6 & \frac{1}{210}\tau_f^7 \\
0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{6}\tau_f^3 & \frac{1}{12}\tau_f^4 & \frac{1}{20}\tau_f^5 & \frac{1}{30}\tau_f^6 \\
0 & 0 & 1 & \tau_f & \frac{1}{2}\tau_f^2 & \frac{1}{3}\tau_f^3 & \frac{1}{4}\tau_f^4 & \frac{1}{5}\tau_f^5 \\
0 & 0 & 0 & 1 & \tau_f & \tau_f^2 & \tau_f^3 & \tau_f^4
\end{bmatrix}
\begin{bmatrix}
a_{i0} \\ a_{i1} \\ a_{i2} \\ a_{i3} \\ a_{i4} \\ a_{i5} \\ a_{i6} \\ a_{i7}
\end{bmatrix}
=
\begin{bmatrix}
x_{i0} \\ x_{i0}' \\ x_{i0}'' \\ x_{i0}''' \\ x_{if} \\ x_{if}' \\ x_{if}'' \\ x_{if}'''
\end{bmatrix}
\tag{123}
$$

If these equations are resolved for the coefficients a_{ik} in Matlab will yield [64]:

$$
a_{i0} = x_{i0}, \quad a_{i1} = x_{i0}', \quad a_{i2} = x_{i0}'', \quad a_{i3} = x_{i0}'''
$$

$$
a_{i4} = \frac{-4x_{if}''' + 16x_{i0}'''}{\tau_f} + \frac{60x_{if}'' - 120x_{i0}''}{\tau_f^2} + \frac{-360x_{if}' - 480x_{i0}'}{\tau_f^3} + \frac{840x_{if} - 840x_{i0}}{\tau_f^4}
$$

$$
a_{i5} = \frac{30x_{if}''' + 60x_{i0}'''}{\tau_f^2} + \frac{-420x_{if}'' + 600x_{i0}''}{\tau_f^3} + \frac{2340x_{if}' - 2700x_{i0}'}{\tau_f^4} + \frac{-5040x_{if} + 5040x_{i0}}{\tau_f^5} \tag{124}
$$

$$
a_{i6} = \frac{-60x_{if}''' - 80x_{i0}'''}{\tau_f^3} + \frac{780x_{if}'' - 900x_{i0}''}{\tau_f^4} + \frac{-4080x_{if}' - 4320x_{i0}'}{\tau_f^5} + \frac{8400x_{if} - 8400x_{i0}}{\tau_f^6}
$$

$$
a_{i7} = \frac{35x_{if}''' + 35x_{i0}'''}{\tau_f^4} + \frac{-420x_{if}'' + 420x_{i0}''}{\tau_f^5} + \frac{2100x_{if}' + 2100x_{i0}'}{\tau_f^6} + \frac{-4200x_{if} + 4200x_{i0}}{\tau_f^7}
$$

and the final arc τ_f becomes the first optimization and varied parameter.

5.   Time and Space Decoupling

Since the speed is related to the Cartesian coordinates by the equation [71]:

$$V(t) = \sqrt{\dot{X}(t)^2 + \dot{Y}(t)^2 + \dot{Z}(t)^2}$$

we also specify a speed profile along the trajectory by using time t, as an argument, ending to define the trajectory itself, too.

In order to decouple the trajectory from the speed profile we can utilize the abstract argument τ which connects to time through the variable speed factor

$$\lambda(\tau) = \frac{d\tau}{dt} \tag{125}$$

By this way, we manage to vary the speed profile along the same trajectory by changing the speed factor $\lambda(\tau)$ [71]:

$$V(\tau) = \lambda(\tau)\sqrt{X'(\tau)^2 + Y'(\tau)^2 + Z'(\tau)^2} \tag{126}$$

For the quadrotor's case though, it is more useful to approximate the speed factor with the same procedure as for the reference trajectory before in order to achieve parameterizing the speed factor so as afterwards compute the speed profile.

So, if we assume that:

$$
\begin{aligned}
\lambda(\tau) &= \sum_{k=0}^{n} a_k^{\lambda} \frac{(\max(1, k-2))!\tau^k}{k!} \\
\lambda'(\tau) &= \sum_{k=1}^{n} a_k^{\lambda} \frac{(\max(1, k-2))!\tau^{k-1}}{(k-1)!} \\
\lambda''(\tau) &= \sum_{k=2}^{n} a_k^{\lambda} \tau^{k-2} \\
\lambda'''(\tau) &= \sum_{k=3}^{n} (k-2) a_k^{\lambda} \tau^{k-3}
\end{aligned}
\tag{127}
$$

And for n=5:

$$\lambda(\tau) = P_\lambda(\tau) = \sum_{k=0}^{5} a_k^\lambda \tau^k = a_0^\lambda + a_1^\lambda \tau^1 + a_2^\lambda \tau^2 + a_3^\lambda \tau^3 + a_4^\lambda \tau^4 + a_5^\lambda \tau^5 \qquad (128)$$

where coefficients are the solutions of:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & \tau_f & \dfrac{1}{2}\tau_f^2 & \dfrac{1}{6}\tau_f^3 & \dfrac{1}{12}\tau_f^4 & \dfrac{1}{20}\tau_f^5 \\ 0 & 1 & \tau_f & \dfrac{1}{2}\tau_f^2 & \dfrac{1}{3}\tau_f^3 & \dfrac{1}{4}\tau_f^4 \\ 0 & 0 & 1 & \tau_f & \tau_f^2 & \tau_f^3 \end{bmatrix} \begin{bmatrix} a_0^\lambda \\ a_1^\lambda \\ a_2^\lambda \\ a_3^\lambda \\ a_4^\lambda \\ a_5^\lambda \end{bmatrix} = \begin{bmatrix} V_0 \\ V_0' \\ \lambda_0'' \\ V_f \\ V_f' \\ \lambda_f'' \end{bmatrix} \qquad (129)$$

Then, finally, the speed profile will be computed as:

$$V(\tau) = P_\lambda(\tau)\sqrt{(P_x'(\tau))^2 + (P_y'(\tau))^2 + (P_z'(\tau))^2} \qquad (130)$$

98

6.   Qball-X4 Inverse Dynamics

a.   *Differential Flatness*

Suppose the dynamics of a system are described by a set of ordinary non-linear differential equations:

$$\dot{x} = f(x, u) \qquad (131)$$

where $x(t) \in X \subset \mathbb{R}^n$, is the state vector and $u(t) \in U \subset \mathbb{R}^n$ is the control vector, and f is some vector function.

By definition from [68], "the differential flatness is the expression of the state and control vectors in terms of some output vector, y." (Without loss of generality we assume y to be a subset of x, i.e. $y = Cx$, where C is a k-by-n matrix, k < n). Also from [69], "For a system to be differentially flat and therefore possess a flat output it requires a set of variables $y(t) = Cx \in Y \subset \mathbb{R}^k$ such that:

- The components of **y** are not differentially related over $\hat{A}$;

- Every state and control may be expressed as a function of the output vector **y** and a finite number of their time derivatives, i.e.

$$x = h_1(y, \dot{y}, \ddot{y}, \dddot{y}, ...) \quad \text{and} \quad u = h_2(y, \dot{y}, \ddot{y}, \dddot{y}, ...)$$

which is essentially the inversion of the original system $\dot{x} = f(x, u)$ with respect to the output vector $y = Cx$."

b.   *Quadrotor's Inverse Dynamics*

Attempting to address the differential flatness property in quadrotor's dynamics, the output vector will consist by four components, since we have four controls. These components will be the translational positions x, y, z, and the yaw angle ψ, as it can be dynamically decoupled from the other states [70] (in case the control input U is used to set the yaw angle to zero).

So, the output vector Y can be defined as:

$$Y = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^T = \begin{bmatrix} I_{3x3} & 0_{3x5} & 0 & 0_{3x3} \\ 0 & 0_{1x5} & 1 & 0_{1x3} \end{bmatrix} X = CX \qquad (132)$$

Now, the control inputs can be expressed as a function of the states and their derivatives as follows:

$$U_1 = \sqrt{\ddot{x}^2 + \ddot{y}^2 + (g + \ddot{z})^2}$$
$$U_2 = \ddot{\phi}$$
$$U_3 = \ddot{\theta} \tag{133}$$
$$U_4 = \ddot{\psi}$$

We have to take the second derivative of the states ϕ, θ, the rotational part of the state vector, in order to express all the control inputs toward output vector. So, in order to express those states in terms of the output vector, we follow the procedure below:

If we take individually the three equations of Eq. 90 and by rearranging and multiplying the third one with "tanθ" we have:

$$\ddot{x} = U_1 \sin\phi \sin\psi + U_1 \cos\phi \sin\theta \cos\psi$$
$$\ddot{y} = -U_1 \sin\phi \cos\psi + U_1 \cos\phi \sin\theta \sin\psi$$
$$(\ddot{z} + g)\tan\theta = U_1 \cos\phi \sin\theta$$

If we substitute the third equation to the other two, we get:

$$\left.\begin{array}{l} \ddot{x} = U_1 \sin\phi \sin\psi + (g + \ddot{z})\tan\theta \cos\psi \\ \ddot{y} = -U_1 \sin\phi \cos\psi + (g + \ddot{z})\tan\theta \cos\psi \end{array}\right\}\begin{array}{l}\times\cos\psi\\ \\ \times\sin\psi\end{array} \Rightarrow$$

$$\Rightarrow \left.\begin{array}{l} \ddot{x}\cos\psi = U_1 \sin\phi \sin\psi \cos\psi + (g + \ddot{z})\tan\theta \cos^2\psi \\ \ddot{y}\sin\psi = -U_1 \sin\phi \sin\psi \cos\psi + (g + \ddot{z})\tan\theta \sin^2\psi \end{array}\right\}\begin{array}{l}(+)\\ \Rightarrow\end{array}$$

$$\Rightarrow \ddot{x}\cos\psi + \ddot{y}\sin\psi = (g + \ddot{z})\tan\theta \cos^2\psi + (g + \ddot{z})\tan\theta \sin^2\psi$$

$$\Rightarrow \ddot{x}\cos\psi + \ddot{y}\sin\psi = (g + \ddot{z})\tan\theta \underline{(\cos^2\psi + \sin^2\psi)}^{1}$$

$$\Rightarrow \tan\theta = \frac{\ddot{x}\cos\psi + \ddot{y}\sin\psi}{g + \ddot{z}}$$

$$\Rightarrow \theta = \arctan\left[\frac{\ddot{x}\cos\psi + \ddot{y}\sin\psi}{g + \ddot{z}}\right] \tag{134}$$

100

With the same procedure if we multiply with the opposite way and subtract the two equations we get:

$$\left.\begin{array}{l} \ddot{x} = U_1 \sin\phi \sin\psi + (g+\ddot{z})\tan\theta\cos\psi \\ \ddot{y} = -U_1 \sin\phi\cos\psi + (g+\ddot{z})\tan\theta\cos\psi \end{array}\right\} \begin{array}{l} \times\sin\psi \\ \times\cos\psi \end{array} \Rightarrow$$

$$\Rightarrow \left.\begin{array}{l} \ddot{x}\sin\psi = U_1 \sin\phi\sin^2\psi + (g+\ddot{z})\tan\theta\sin\psi\cos\psi \\ \ddot{y}\cos\psi = -U_1 \sin\phi\cos^2\psi + (g+\ddot{z})\tan\theta\sin\psi\cos\psi \end{array}\right\} \begin{array}{l}(-)\end{array} \Rightarrow$$

$$\Rightarrow \ddot{x}\sin\psi - \ddot{y}\cos\psi = U_1 \sin\phi\sin^2\psi + U_1\sin\phi\cos^2\psi$$

$$\Rightarrow \ddot{x}\sin\psi - \ddot{y}\cos\psi = U_1 \sin\phi \underbrace{(\cos^2\psi+\sin^2\psi)}^{1}$$

$$\Rightarrow \sin\phi = \frac{\ddot{x}\sin\psi - \ddot{y}\cos\psi}{U_1}$$

$$\Rightarrow \phi = \arcsin\left[\frac{\ddot{x}\sin\psi - \ddot{y}\cos\psi}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (g+\ddot{z})^2}}\right] \tag{135}$$

Having the state vector components expressed in terms of the output vector, we proceed to do so for their derivatives also are:

$$\dot{\theta} = \frac{(\dddot{x}\cos\psi + \dddot{y}\sin\psi)(g+\ddot{z}) - (\ddot{x}\cos\psi + \ddot{y}\sin\psi)\dddot{z}}{(g+\ddot{z})^2 + (\ddot{x}\cos\psi + \ddot{y}\sin\psi)^2} \Rightarrow$$

$$\dot{\theta} = \frac{(\dddot{x}\cos\psi + \dddot{y}\sin\psi)(g+\ddot{z}) - (g+\ddot{z})(\tan\theta\dddot{z})}{(g+\ddot{z})^2 + (\ddot{x}\cos\psi + \ddot{y}\sin\psi)^2} \Rightarrow$$

$$\dot{\theta} = \frac{(\dddot{x}\cos\psi + \dddot{y}\sin\psi - \tan\theta\dddot{z})(g+\ddot{z})}{(g+\ddot{z})^2 + (\ddot{x}\cos\psi + \ddot{y}\sin\psi)^2} \tag{136}$$

and

101

$$\dot{\phi} = \frac{(\ddot{x}\sin\psi - \ddot{y}\cos\psi)U_1 - (\ddot{x}\sin\psi - \ddot{y}\cos\psi)\dot{U}_1}{U_1\sqrt{U_1^2 - (\ddot{x}\sin\psi - \ddot{y}\cos\psi)^2}} \Rightarrow$$

$$\dot{\phi} = \frac{(\ddot{x}\sin\psi - \ddot{y}\cos\psi)U_1 - U_1\sin\phi\dot{U}_1}{U_1\sqrt{U_1^2 - (\ddot{x}\sin\psi - \ddot{y}\cos\psi)^2}} \Rightarrow$$

$$\Rightarrow \dot{\phi} = \frac{(\ddot{x}\sin\psi - \ddot{y}\cos\psi) - \dot{U}_1\sin\phi}{\sqrt{U_1^2 - (\ddot{x}\sin\psi - \ddot{y}\cos\psi)^2}} \tag{137}$$

The second order derivatives:

$$\ddot{\theta} = (g+\ddot{z})\left[\frac{(\dddot{x}\cos\psi + \dddot{y}\sin\psi - \tan\theta\dddot{z}) - 2\dot{\theta}\left[\dddot{z} + (\ddot{x}\cos\psi + \ddot{y}\sin\psi)\tan\theta\right]}{\left[(g+\ddot{z})^2 + (\ddot{x}\cos\psi + \ddot{y}\sin\psi)^2\right]} + \dddot{z}\dot{\theta}\right]$$

$$\ddot{\phi} = \frac{\left[(\dddot{x}\sin\psi - \dddot{y}\cos\psi) - \ddot{U}_1\sin\phi\right]}{\sqrt{U_1^2 - (\ddot{x}\sin\psi - \ddot{y}cos\psi)^2}} - \frac{\dot{\phi}\left[\dot{U}_1 - \sin\phi(\ddot{x}\sin\psi - \ddot{y}\cos\psi)\right]U_1}{U_1^2 - (\ddot{x}\sin\psi - \ddot{y}\cos\psi)^2} \tag{138}$$

Instead of using the exact values, we could also employ central difference approximations

$$\ddot{\theta}_i = \frac{\theta_{i+1} - 2\theta_i - \theta_{i-1}}{\Delta t^2}$$

$$\ddot{\phi}_i = \frac{\phi_i - 2\phi_i - \phi_{i-1}}{\Delta t^2} \tag{139}$$

(forward and backward approximations would be used then for the first and last points).

Finally, we computed the states ϕ, θ in terms of the output vector components, since U_1 is already expressed in terms of x, y, z. When the vehicle is in free fall, singularities may occur, since $g = -\ddot{z}$. One way to avoid it is by constraining the input such that $U_1 > 1$ and the pitch and roll such that $\theta < 90°$ and $\phi < 90°$ as stated in [48]. The differential property of the system provides us with the opportunity to transfer the optimization from the control space to the Output space.

7.   Discretization

As in any numerical method we will compute parameters of the systems in a finite number of points N placed equidistantly along the virtual arc τ_f (varied parameter), thus by dividing the virtual arc τ_f into N–1 equal pieces as shown in Figure so as [71]:

$$\Delta \tau = \tau_f (N-1) \tag{140}$$
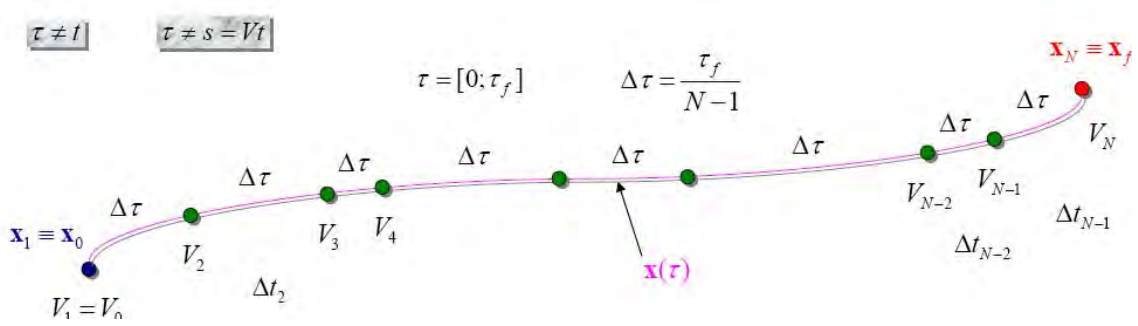
Figure 52.   Excluding Time and Converting Back to Time. After [71]

We then have N equidistant nodes $j = 1,..., N$. All states and controls at the first point $j = 1$ (corresponding to $\tau_1 = \tau_0 = 0$) are defined. For each of the subsequent N–1 nodes $j = 2,.., N,$ the corresponding instants of time, however, need to be computed as follows [71]:

$$\tau_j = (j-1)\Delta \tau \Rightarrow \Delta \tau = \frac{\tau_j}{j-1} \tag{141}$$

$$\Delta t_{j-1} = \frac{2\Delta \tau}{\lambda_{j-1} + \lambda_j} \tag{142}$$

$$t_j = t_{j-1} + \Delta t_{j-1} \tag{143}$$

So, the mapping between the τ and t domains are defined by the speed factor profile P_λ. We can now proceed with computing the states and controls in all nodes. We compute the current value of three Cartesian coordinates and speed using the

corresponding polynomials: $X_j = P_X(\tau_j), Y_j = P_Y(\tau_j)$ and $Z_{j_-}P_Z(\tau_j)$. Then, using the Eq. 141 we compute the time passed since the last sample:

$$\Delta t_{j-1} = 2\frac{\sqrt{(X_j - X_{j-1})^2 + (Y_j - Y_{j-1})^2(Z_j - Z_{j-1})^2}}{(V_j - V_{j-1})} \tag{144}$$

The current time then from Eq.143 equals to $t_j = t_{j-1} + \Delta t_{j-1}(t_1 = t_0 = 0)$, and therefore the current value of the speed factor

$$\lambda_j = \frac{\Delta \tau}{\Delta t_{j-1}} \tag{145}$$

Next, we convert τ derivatives to time derivatives using chain rule relations:

$$\dot{x} = \frac{dx}{dt} = \frac{dx}{d\tau}\frac{d\tau}{dt} = x'\lambda \quad and \quad \ddot{x} = \frac{d(x'\lambda)}{dt} = (x''\lambda + x'\dot{\lambda})\lambda \text{, so that we finally have:}$$

$$\dot{x}_j = \lambda_j x'_j$$
$$\ddot{x}_j = \lambda_j(\lambda'_j x'_j + \lambda_j x''_j)$$
$$\dddot{x}_j = \lambda_j[(\lambda'^2_j + \lambda_j \lambda''_j)x'_j + \lambda_j(3\lambda'_j x''_j + \lambda_j x'''_j)]$$

$$\dot{y}_j = \lambda_j y'_j$$
$$\ddot{y}_j = \lambda_j(\lambda'_j y'_j + \lambda_j y''_j)$$
$$\dddot{y}_j = \lambda_j[(\lambda'^2_j + \lambda_j \lambda''_j)y'_j + \lambda_j(3\lambda'_j y''_j + \lambda_j y'''_j)]$$

$$\dot{z}_j = \lambda_j z'_j$$
$$\ddot{z}_j = \lambda_j(\lambda'_j z'_j + \lambda_j z''_j)$$
$$\dddot{z}_j = \lambda_j[(\lambda'^2_j + \lambda_j \lambda''_j)z'_j + \lambda_j(3\lambda'_j z''_j + \lambda_j z'''_j)]$$

$$\ddot{\psi}_{i,j} = \lambda_j(\lambda'_j \psi'_j + \lambda_j \psi''_j)$$

To convert the initial conditions of the states from time derivatives to virtual arc derivatives, we have to use the inverse of the above relations, so as to compute the controls and the remaining states.

8.   Trajectory Optimization

We have already chosen the reference functions for the X, Y, Z, Ψ and λand computed their coefficients, introduced the inverse dynamics to the system and set the boundary conditions. The next step is to proceed with the trajectory optimization through a specific optimization routine.

a.   *Problem Formulation in the Control Space*

Normally, in order to determine the optimal trajectory, the optimization procedure take place within the control space regarding the applied constraints like state constraints, actuator (control) constraints and obstacle avoidance constraints within the output space and the state space. The problem can be set up as [48]:

$$\min_{U(t)\in U\subset\Re^4} \Phi \qquad \text{for } t \in \left[0,t_f\right] \text{ such that}$$

$$\dot{Y}-Cg(X,U)=0$$
$$\dot{Y}_0-Cg(X_0,U_0)=0$$
$$\dot{Y}_f-Cg(X_f,U_f)=0$$
$$C(X,U)\leq 0 \tag{145}$$

where $\Phi(X,U)$ is the cost function, the initial and final constraints on the states are set according to Eq.132 and Eq.88 at $t=0$ and $t=t_f$, respectfully, the dynamic inequality constraints on the trajectory (for obstacle avoidance) and on the states and inputs (to avoid singularities and to provide constraints on the control signals) are expressed through the set of functions C(X,U).

b.   *Cost Function*

The cost function, Φ, is a quantitative measure of the optimality of the trajectory and consists by the sum of two components, the running costs and the terminal cost. If the running costs (battery consumption) are proportional to the average velocity, the objective function can be defined as [48]:

$$\Phi = (1-w)\frac{1}{t}\int_0^{t_f}\sqrt{P_1\dot{x}^2 + P_1\dot{y}^2 + P_1\dot{z}^2}\,dt + w(t_f - T)^2 \qquad (146)$$

where w, P_1, P_2, P_3 are the weighting factors (not necessarily equal to each other), and T is the predetermined time of arrival. The minimum-time case takes place when w=1 and T=0 and the minimum-fuel case when w=0 and $P_1 = P_2 = P_3$. So, it is clear that the mission scenario will determine what the cost function is and how many weighting factors have to be adjusted.

Having proved through differential flatness that the state vector x and the control vector u can be both expressed via the derivatives of the output vector y

$$X = h_1(Y), U = h_2(Y) \qquad (147)$$

We can reformulate the optimization problem in the output space.

c.   *Problem Formulation in the Output Space*

If optimization takes place within the output space (differential flatness properties) as opposed to the control space, it would be very useful because the constraints occurring for example from obstacle avoidance happens in the output space, hence the computation time for constraint handling is reduced drastically. As opposed to (145) the problem can now be reformulated as follows:

$$\min_{U(t)\in U\subset\Re^4}\Phi \qquad \text{for } t\in\left[0,t_f\right]\text{ such that:}$$

$$\dot{Y}_0 - g^*(Y(0)) = 0$$
$$\dot{Y}_f - g^*(t_f) = 0$$
$$C^*(Y) \le 0 \qquad (148)$$

where $\quad g^*(Y) = Cg(X,U) = Cg(h_1(Y), h_2(Y))\ C^*(Y) = C(X,U) = C(h_1(Y), h_2(Y))$ come from Eq. 147. With proper parameterization this problem can be solved in MATLAB using the optimization toolbox function fmincon.

d.   *Parameterization*

In order to reduce the dimension of the problem to a finite amount, it is suggested that the three translational outputs (x, y and z) be parameterized (the fourth output, the yaw angle ψ, is assumed to be zero).

Thus   the   equations   derived   so   far   will   become:

$$\theta = \arctan\left[\frac{\ddot{x}}{g + \ddot{z}}\right] \tag{149}$$

$$\phi = \arcsin\left[\frac{\ddot{y}}{\sqrt{\ddot{x}^2 + \ddot{y}^2 + (g + \ddot{z})^2}}\right] \tag{150}$$

The derivatives also are:

$$\dot{\theta} = \frac{\dddot{x}(g + \ddot{z}) - \ddot{x}(\dddot{z})}{(g + \ddot{z})^2 + \ddot{x}^2} \quad \text{or} \quad \dot{\theta} = \frac{(\dddot{x} - \tan\theta\,\dddot{z})(g + \ddot{z})}{(g + \ddot{z})^2 + \ddot{x}^2} \tag{151}$$

$$\dot{\phi} = \frac{-\dddot{y}\sqrt{\ddot{x}^2 + \ddot{y}^2 + (g + \ddot{z})^2} + \ddot{y}(\ddot{x}\dddot{x} + \ddot{y}\dddot{y} + \ddot{z}(g + \ddot{z}))}{\left[\ddot{x}^2 + \ddot{y}^2 + (g + \ddot{z})^2\right]\sqrt{\ddot{x}^2 + (g + \ddot{z})^2}} \quad \text{or}$$

$$\dot{\phi} = \frac{-\dddot{y}U_1 + \ddot{y}\dot{U}_1}{U_1\sqrt{U_1^2 - \ddot{y}^2}} = \frac{-\dddot{y} - \dot{U}_1 \sin\phi}{\sqrt{U_1^2 - \ddot{y}^2}} \tag{152}$$

The second order derivatives:

$$\ddot{\theta} = (g + \ddot{z})\left[\frac{(\ddddot{x} - \tan\theta\,\ddddot{z}) - 2\dot{\theta}\left[\dddot{z} + (\dddot{x} + \dddot{y})\tan\theta\right]}{\left[(g + \ddot{z})^2 + \ddot{x}^2\right]}\right] + \dddot{z}\dot{\theta} \tag{153}$$

$$\ddot{\phi} = \frac{\left[(-\ddddot{y}) - \ddot{U}_1 \sin\phi\right]}{\sqrt{U_1^2 - \ddot{y}^2}} - \frac{\dot{\phi}\left[\dot{U}_1 - \sin\phi(-\ddddot{y})\right]U_1}{U_1^2 - \ddot{y}^2}$$

THIS PAGE INTENTIONALLY LEFT BLANK

V.   SIMULATION

A.   QUADROTOR MAIN INTERFACE

The Quarc / Simulink top level realization of Quadrotor model and controllers developed are shown in Figure 54. This the main screen you see when you open the quadrotor model in Simulink. It was separated into different components for friendly use by the operator.

Qball-X4 Waypoint controller

Issue waypoints to the Qball for navigation control.

Switch between joystick and closed-loop control using the switches inside the Mode Control subsystem.

In closed-loop flight, control the position of the Qball-X4 by setting height and heading in the Position Commands subsystem.

View IMU data and motor output signals in the HiQ subsystem.
Data is logged to a host MAT-file in HiQ\SAVE DATA (black box)

Bad Link

Position Commands

Mode control

Calculate Roll Pitch Heading Height

SAVE DATA (black box)

HiQ

Joystick from host

Pitch Controller

Roll Controller

Yaw Controller

Control signal mixing
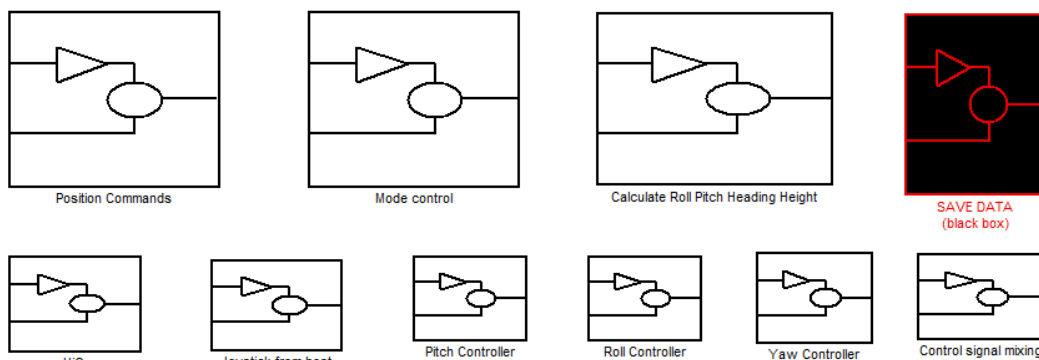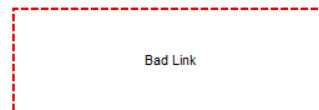
Figure 53.   Simulink Representation of Qball-X4 Controller

The system consists of the following subsystems blocks:

1.   Positions Commands Subsystem

The Position Commands subsystem contains the waypoint state machine described in the previous section, where its outputs are the throttle command, the x, y, z position and the heading commands as shown below in the simulink diagram. The Y axis is called Z inside the optitrack software. So, where Z is meant Y and where height is meant Z.

109
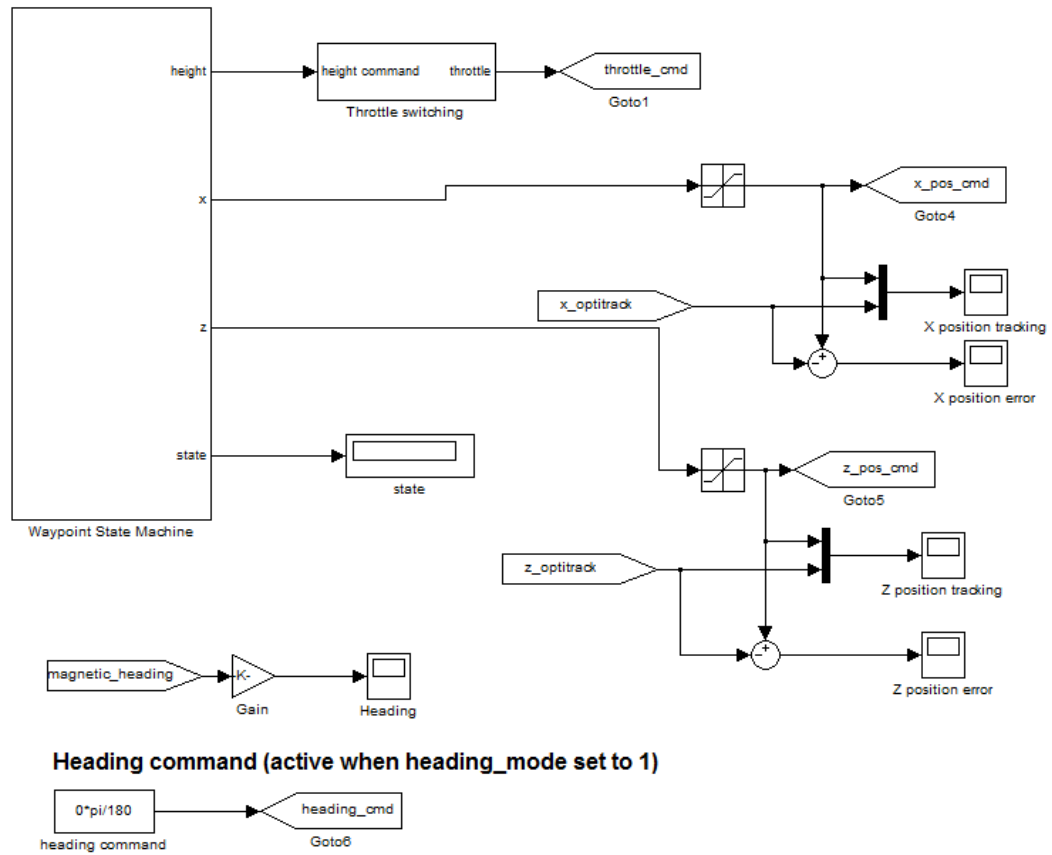
Figure 54.   Positions Commands Subsystem

As far as the heading command is concerned, a stable heading set up to 8 degrees was used for the experiments. This eliminates the small disturbance of the heading when the quadrotor takes off. I assume this phenomenon exists because of the metallic objects inside the lab that cause some interference in the magnetometer measurements.

2.   Mode Control Subsystem

The Mode Control Subsystem allows the operator to select height, position and heading modes for autonomous or 4-channel joystick control as shown in Figure 56.
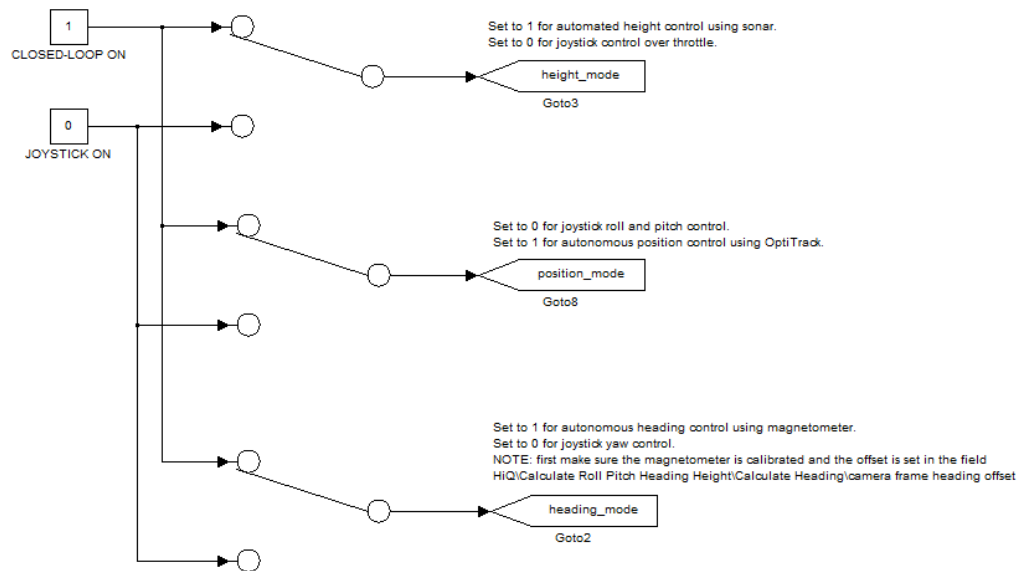
Figure 55.   Mode Control Subsystem

3.   Calculate Roll Pitch Heading Height Subsystem

This subsystem computes the vehicle's orientation through the calculation of roll, pitch, magnetic heading and the x, y magnetometer components of the quadrotor. Complementary filter was used to correct roll and pitch components.
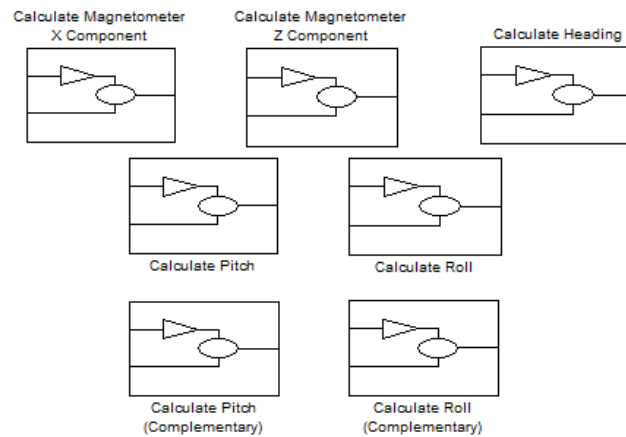
Figure 56.   Calculate Roll Pitch Heading Height Subsystem

4. HiQ Subsystem

The HiQ subsystem consists of the Hardware- In-the-Loop (HIL) blocks used to configure the HiQ acquisition card and read or edit the values. It contains the Motor / PWM outputs that power the whole system and initialize the sensors to get the measurements through the various selectors. The Gain simulink block enables or disables the motors by multiplying the motor input signals by the value of 1 or 0, respectively. Inside this subsystem you can also check the battery voltage.
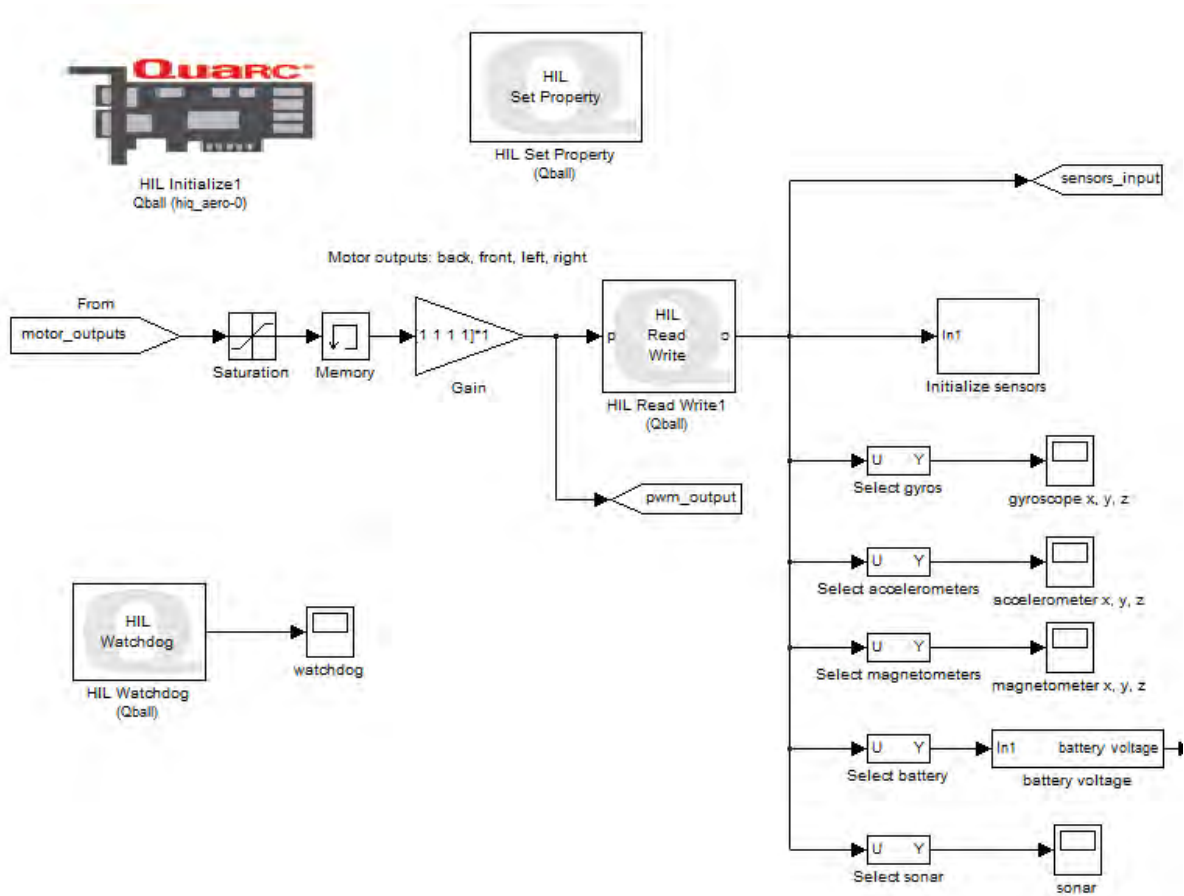
Figure 57.   HiQ Subsystem

5. Joystick from Host Subsystem

This subsystem is an alternative power system to the motors through the 4-channel joystick. It receives streaming data from the host model. If the IP address is set up properly for the ground station in the Stream Client URI, then the Qball is able to connect to the host model. It gives the same commands as in the Position Commands subsystem and receives the data packet from the optitrack system. For safety reasons, if the communications from the host is interrupted for 1 consecutive second, the QBall is ordered to land.
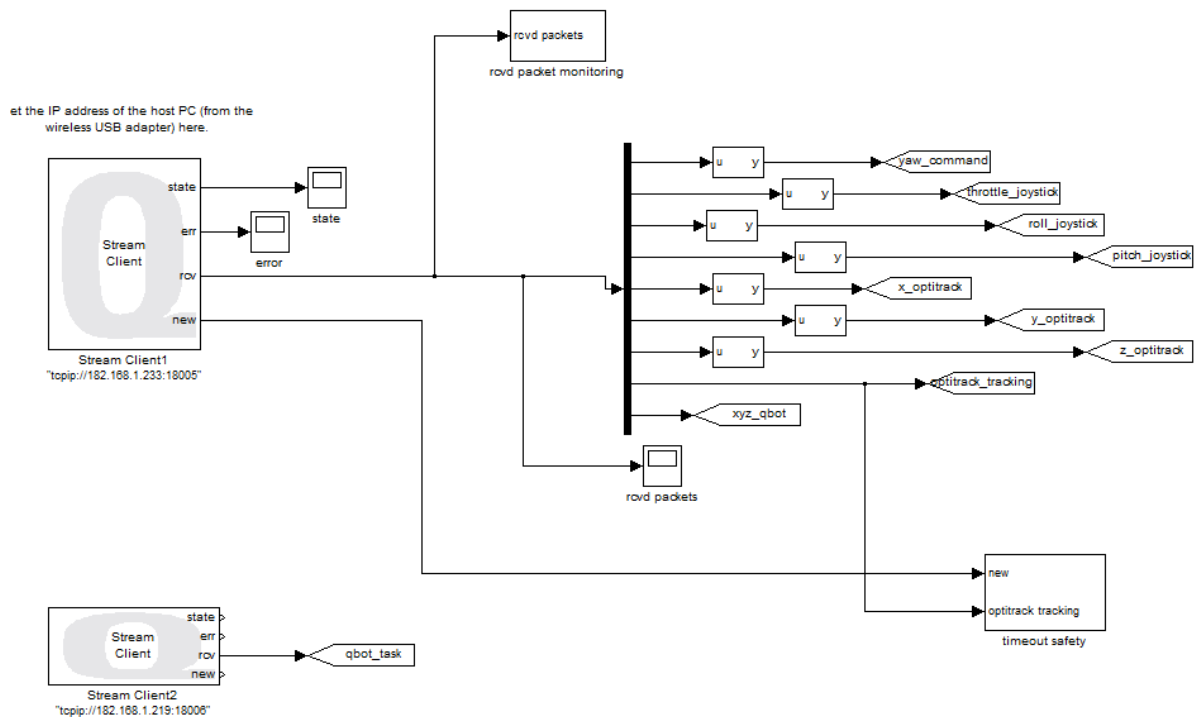
Figure 58.   Joystick from Host Subsystem

6. Save Data Subsystem

The "Save Data" subsystem saves all the data collected from every flight of the quadrotor into a MAT file through the Quarc block "To Host file" for further processing or plotting. Figure 60 shows how all the different signal outputs collected and saved.
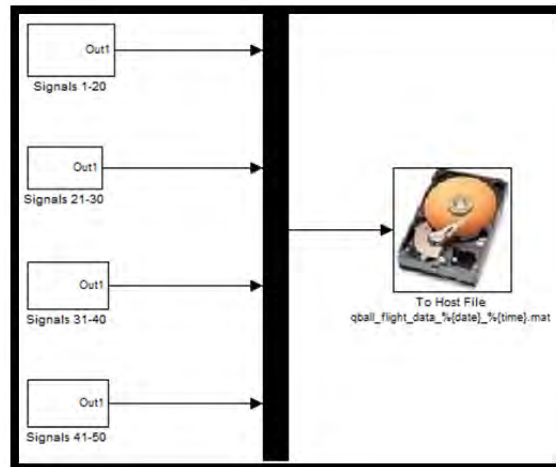
Figure 59.   Save Data Subsystem

B.   QBall X4 WAYPOINT NAVIGATION

1.   Waypoints Input

The waypoints foe the vehicle's navigation will be introduced through the script "Initialize_Qball_Waypoints.m showing in Appendix A, where the user defines the position of the waypoints in the operation X-Y area (again Y axis is represented in the model as Z because of the optitrack system notation). The platform where the waypoints are set is shown below:
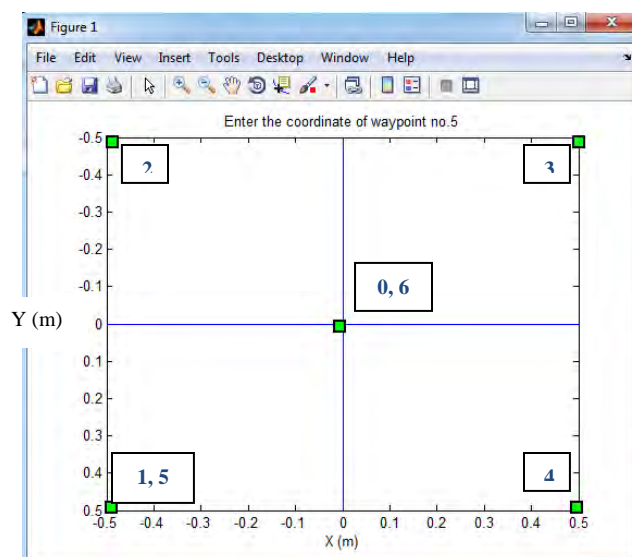
Figure 60.   Waypoints Input Diagram

2.   Waypoint State Machine

The waypoints will be followed through the waypoint state machine shown in Figure 62.
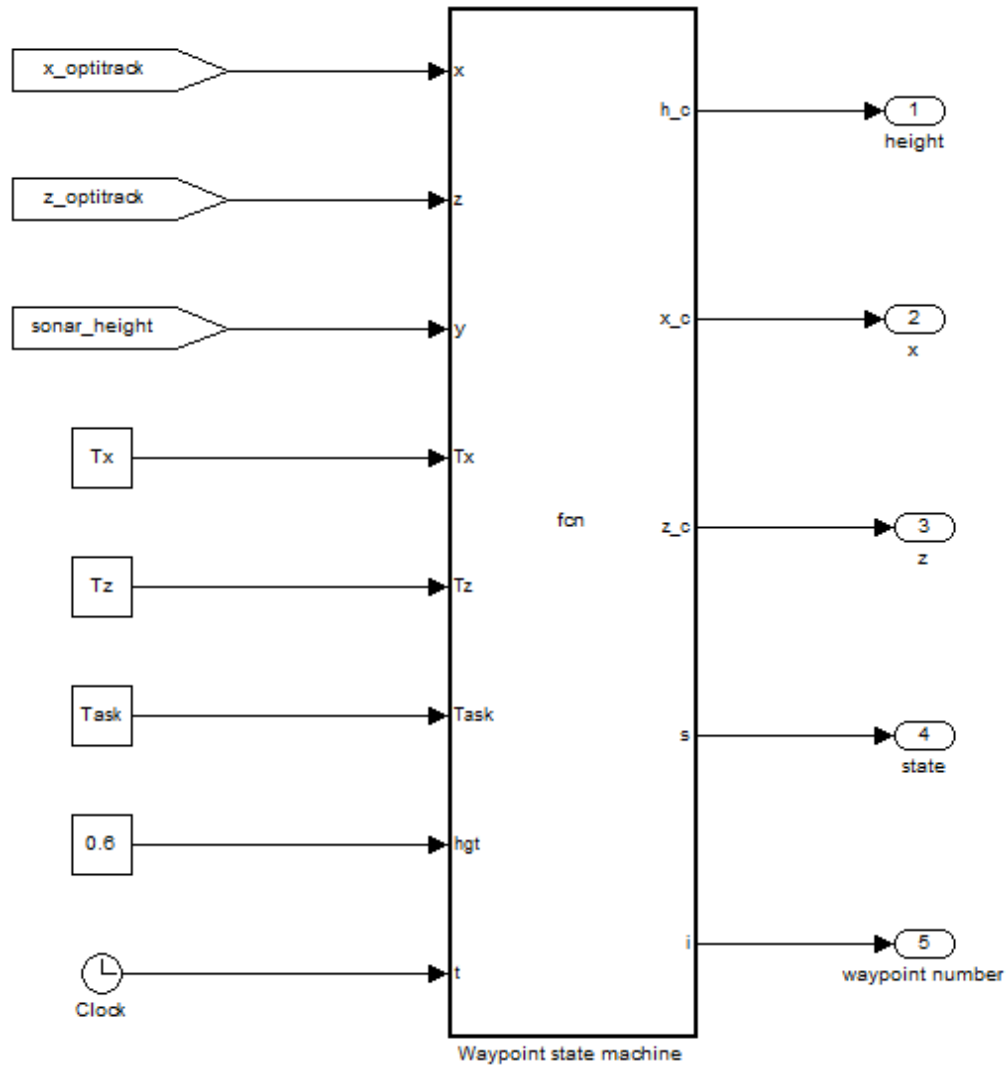
Figure 61.   Waypoint State Machine

The height will be held constant at 0.6 m (hgt coordinate) through the sonar controller. Tx and Tz are the coordinates taken from the previous section and represent the coordinates of x- and y-axis, respectively (This representation came up from the optitrack system set up). The function controls the machine is shown in Appendix A.

3. Waypoint Tracking

The trajectory will be followed through the implemented LQR controller described in the previous chapter and for specific weight factor matrices that will improve the performance. The LQR gains are shown below:

Roll – Pitch Controller Weight Factors: Q= [100 0 22000 10] - R=3000

Output LQR Gains: k(1) = -20.047 + 0.000 i

k(2) = -5.618 + 6.128 i

k(3) = -5.618 - 6.128 i

k(4) = -0.316 + 0.000 i

X-Y Controller Weight factors: Q= [50 2 100 0.1] - R=50

Output LQR Gains: k(1) = 12.080 + 0.000 i

k(2) = -1.762 + 1.604 i

k(3) = 1.762 - 1.604 i

k(4) = -0.045 + 0.000 i

Height Controller Weight factors: Q = diag([1 0 50]) - R = 5000000

Output LQR Gains: k(1) = -0.517 + 0.890 i

k(2) = -0.517 - 0.890i

k(3) = -1.024 + 0.000 i

Yaw Controller Weight factors: Qy = diag([1 0.1]) - Ry = 1000

Output LQR Gains: k(1) = -3.762 + 1.287 i

k(2) = -3.762 -1.287 i

The representation of how accurate is the waypoint tracking is shown in figure 63 (two trajectories). The quadrotor takes off, stabilizes in the assigned altitude and then follows the waypoints 1 to 6 added from the operator (shown in Figure 61), where the starting point is the zero point (0, 0) of the localization system. When each waypoint is

116

found, the quadrotor has to stay 5 sec. A little overshoot takes place because of the acceleration, it has already gained. The smaller the distance the smaller the overshoot. The second trajectory takes place with low battery. Through the experiments, it was shown that as the battery goes off; the response of the quadrotor is degraded.
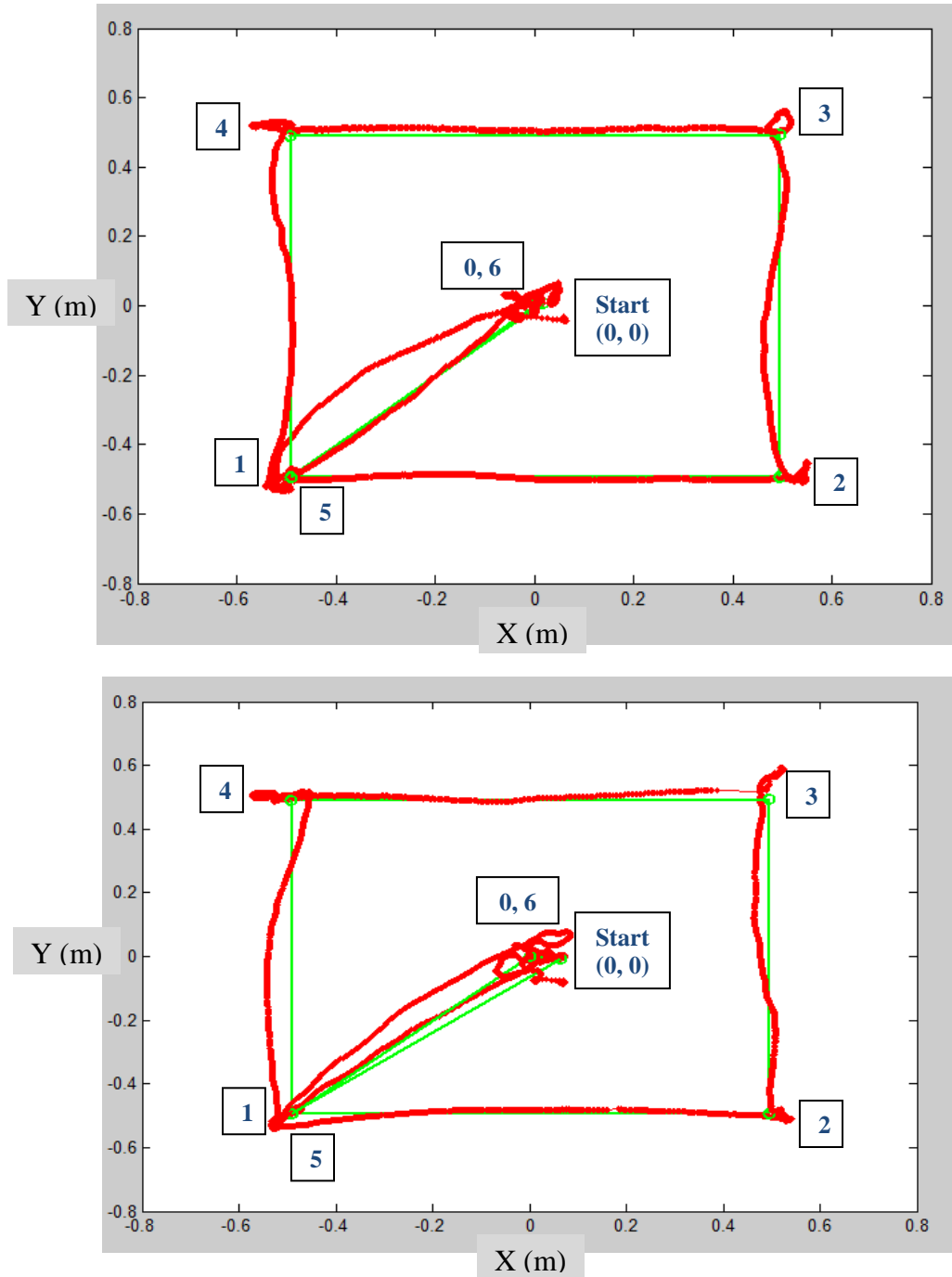
Figure 62.   QBall Actual Trajectories (With Full battery and low battery)

4. Plots

Several plots can be obtained using the Save data interface. Some of them can be really useful for understanding the response of the vehicle and the controller; some others just for testing purposes in order to check if we can obtain data from the quadrotor sensors. The PWM input of every motor toward time is shown Figure 64. The initial value is the 5% of the 20 ms, thus 1 ms that is the minimum throttle and it goes up to 9% of the 20 ms, thus 1.8 ms (maximum throttle=2 ms).
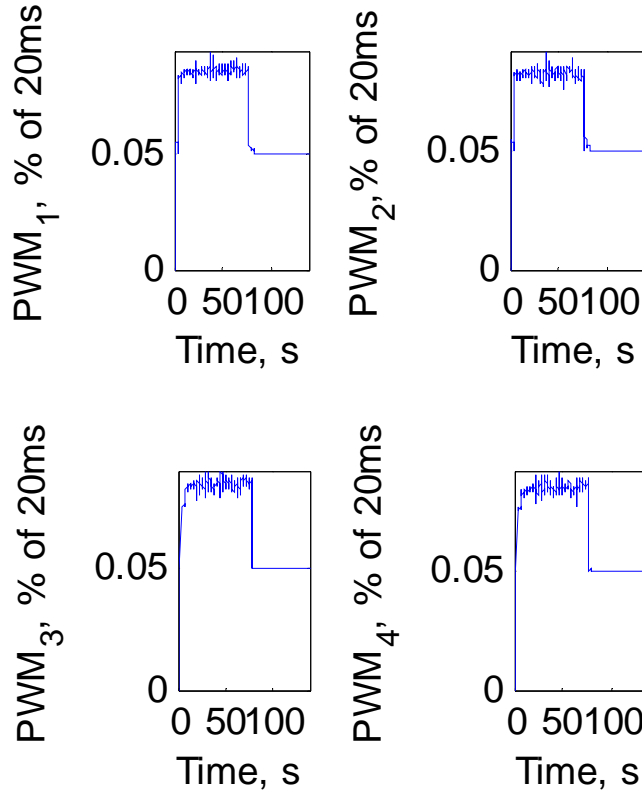
Figure 63.   PWM Input for each motor

The following figures depict the data from each sensor and can be used for testing purposes. Figure 65 shows the gyroscope measurements in rad/s, where small variations around zero take place. Figure 66 shows the accelerometer measurements in m/s. The Z acceleration oscillates around -10, the value of the gravitational acceleration. Figure 67 shows the magnetometer measurements in Gauss, with the Z value to be the largest.

118
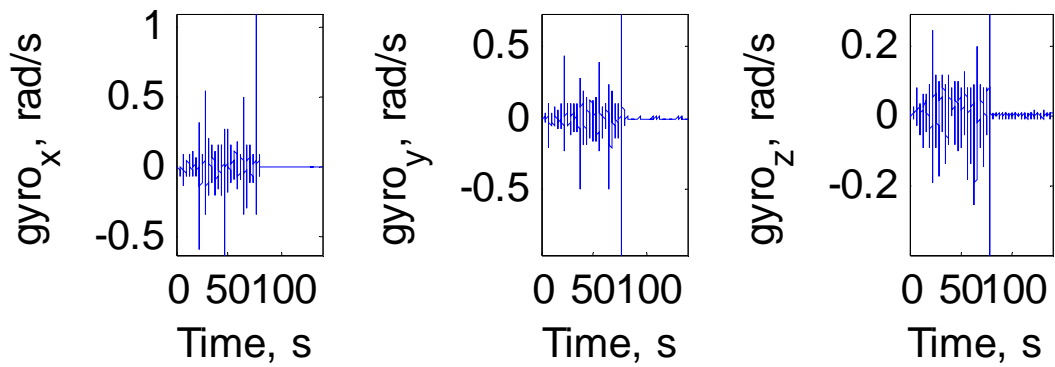
Figure 64.   Gyroscopes measurements
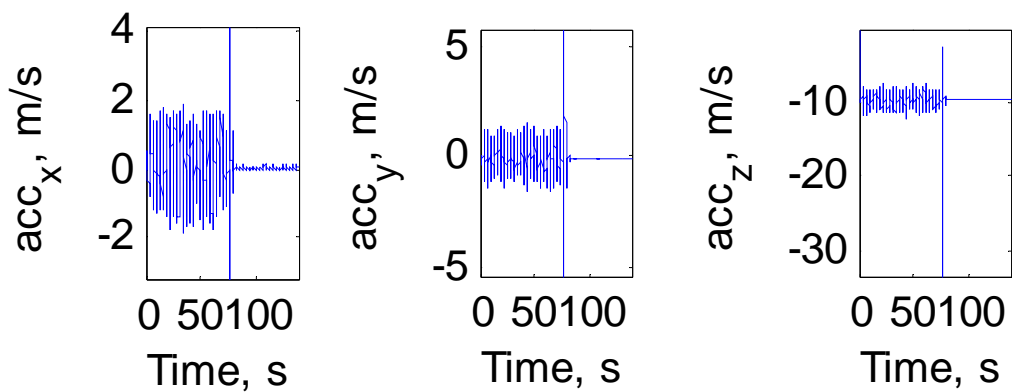
Figure 65.   Accelerometers measurements
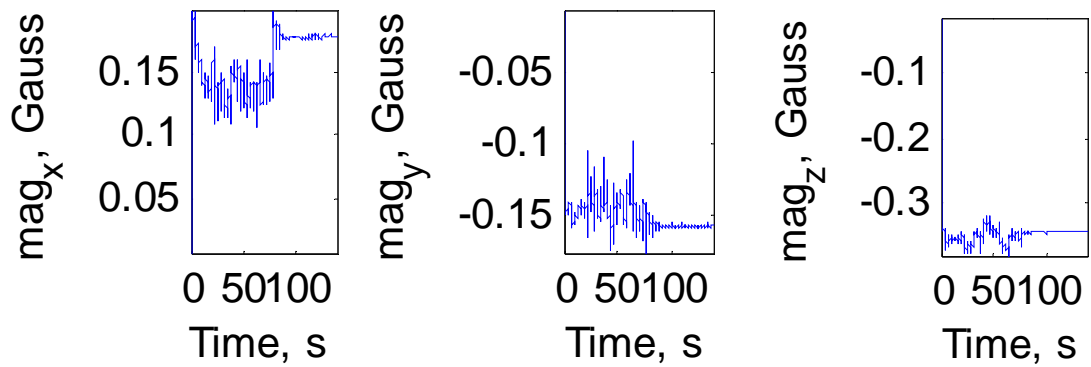
Figure 66.   Magnetometer measurements

The observable and magnetic heading are shown in Figure 68. The magnetic heading is the straight calculation of the atan2 (mag_y / mag_x), where mag_x, mag_y are the measurements of the magnetometer. The observable heading is the corrected one using feedback from the magnetic heading.
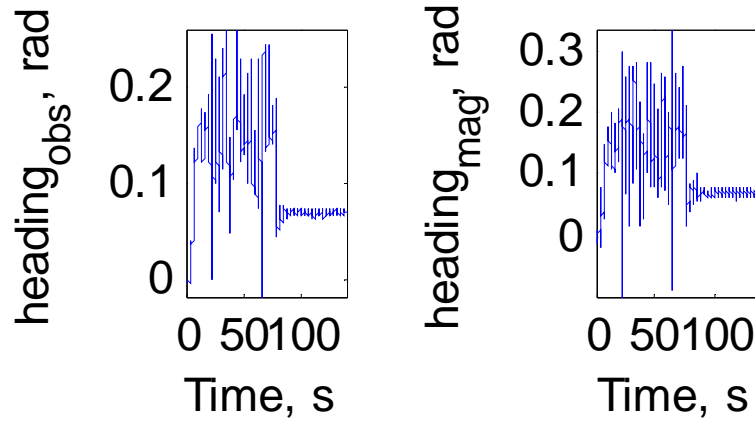
Figure 67.   Observable and magnetic heading

The following plots present the battery voltage and the chosen modes for autonomous navigation. If the mode is 0 the navigation is taking place through the 4 channel joystick, else if the mode is 1 then the navigation is autonomous. The battery voltage drops from flight to flight. if the voltage goes under 10.6V then the battery becomes useless.
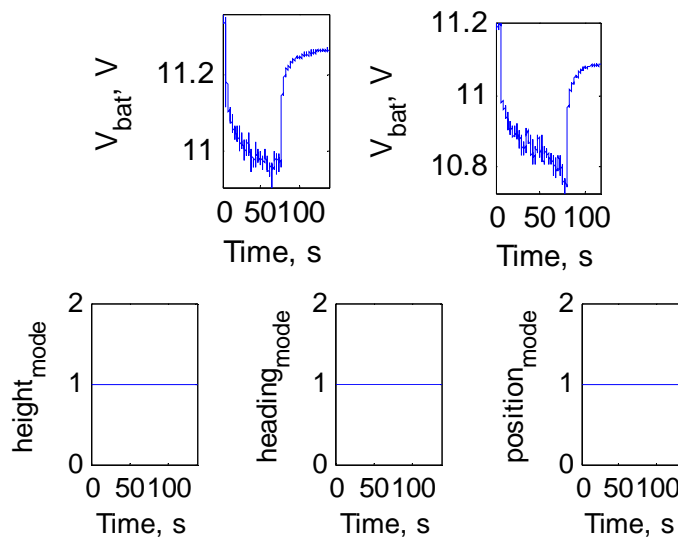
Figure 68.   Battery voltage and height-heading-position mode (auto=1)

The following plots in figures show the comparison of the measured and the commanded X, Y, Z data. The small differences shown is having to do with the disturbances occurred during the flight as well as with the linear controller it was used. The difference in height between measured and command Z is because the optitrack system captures the reflector placed in the top of the quadrotor, thus 0.6 m from the ground.
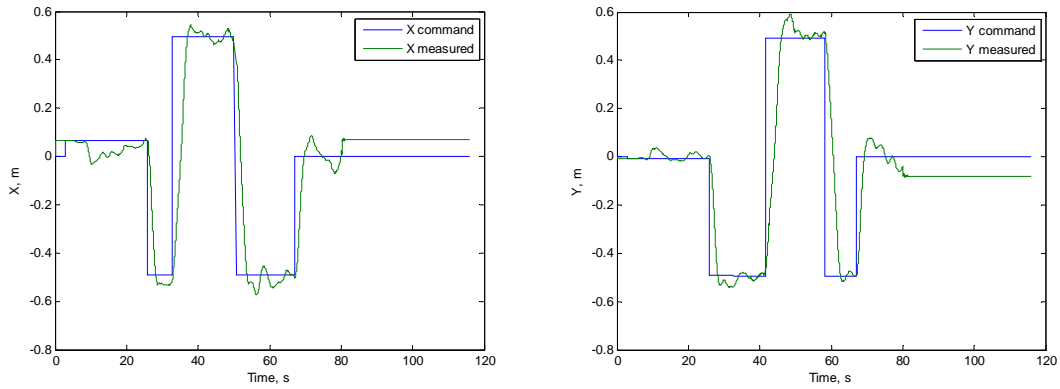
Figure 69. X- Y position comparisons
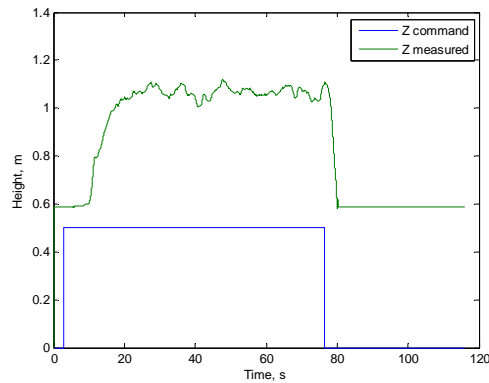
Figure 70. Height position comparison

The comparison of the measured and commanded roll and pitch data is shown below. The difference actually in both roll and pitch response is quite small. The measured roll and pitch values are already filtered, as it is noticed form the graphs. The

121

only distinct difference is happening in the first seconds when the takeoff takes place. It is a phenomenon under investigation since the quadrotor tends to roll a little bit as it takes off and climbing to the assigned height. The roll controller gains should be tested more in order to eliminate this effect.
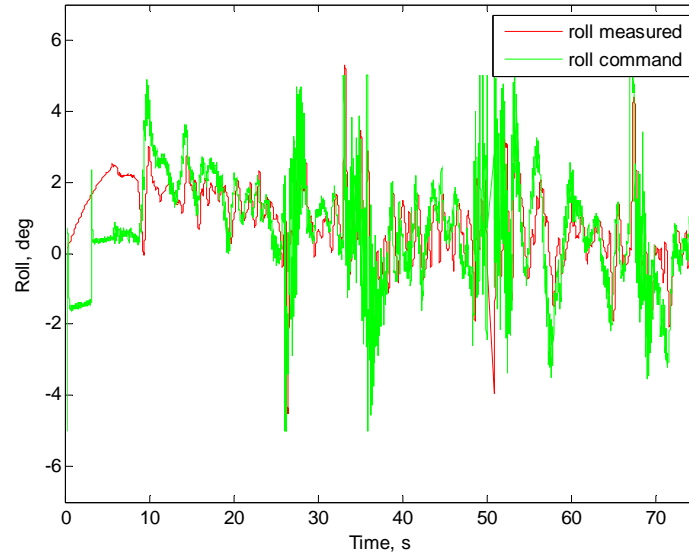
Figure 71.   Roll comparison
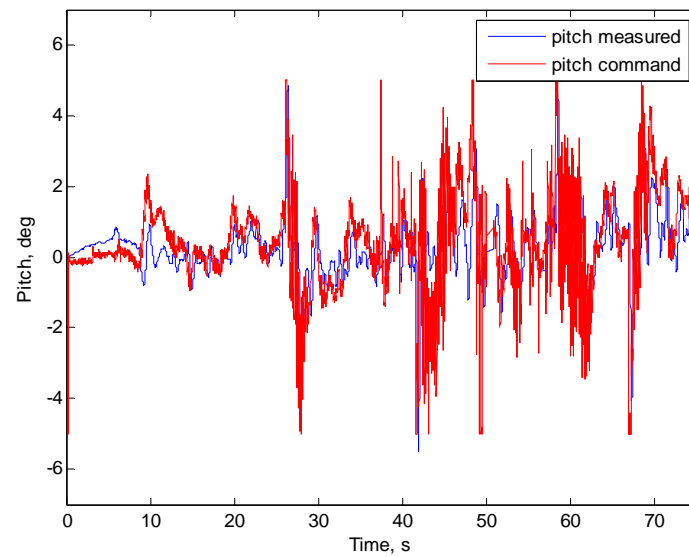
Figure 72.   Pitch comparison

B.   QBOT WAYPOINT NAVIGATION

1.   Waypoints Input

The waypoints will be introduced through the script "Initialize_Qbot_Waypoints.m showing below where the user defines the position of the waypoints in the operation area (x-y area).

As we see in the m- file, there is a calibration file loaded, coming from a magnetic calibration process that took place before the navigation process began, in order to set up the compass to the environment of the laboratory taking into account the declination of Monterey that is different from Toronto (Quanser company location) where the initial calibration was done. The simulink model of the compass and the model that initiates the calibration are shown in Figures 74 and 75, respectively.
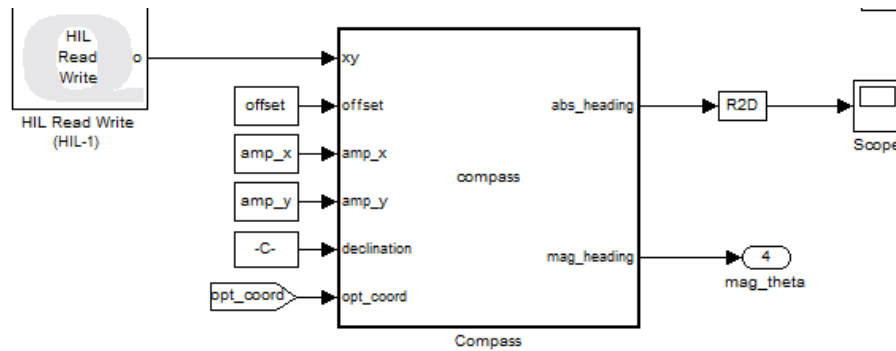
Figure 73.   Compass Model
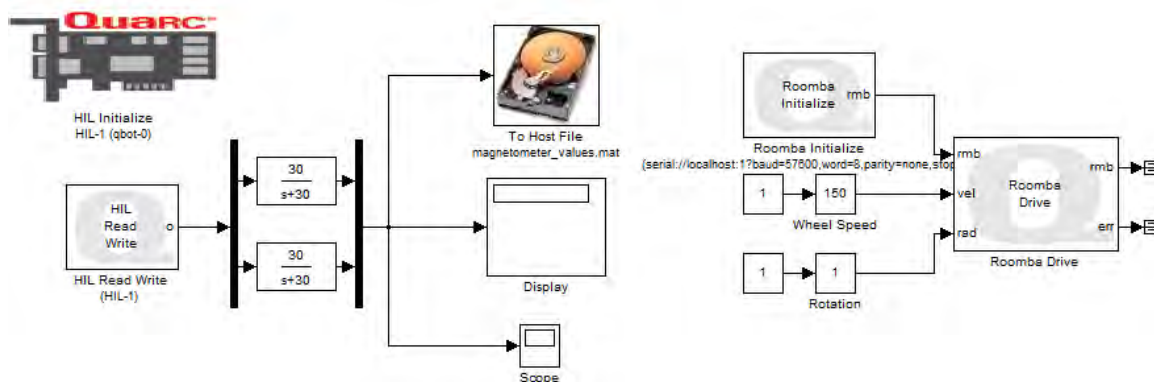
Figure 74.   Qbot_magnetometer_calib model

123

The Calibrated magnetometer x (red), y(green) measurements are shown in Figure 76. The difference in time occurs because of the offset imposed for better representation. The calibration is succeeded when the values come closer to the absolute 1, -1. So the performed one is better for the y measurements.
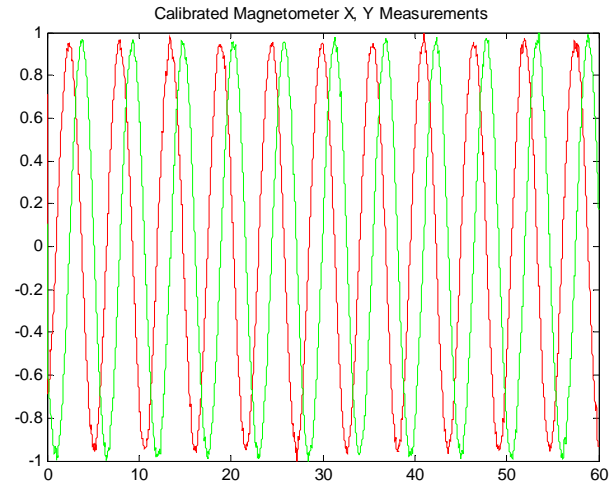
Figure 75.   Calibrated Magnetometer X, Y Measurements

The platform where the waypoints are set are shown in Figure 77. Six waypoints are chosen. The Qbot is placed in the zero point of the localization system.
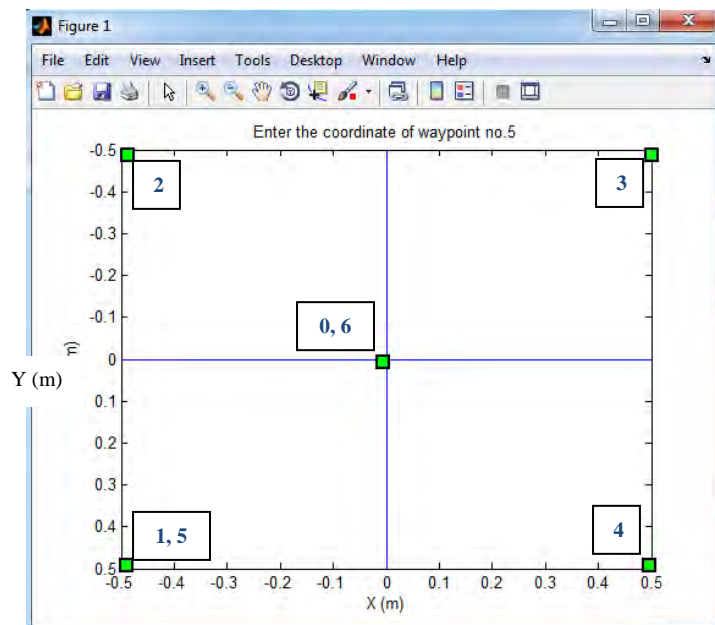
Figure 76.   Waypoints Input

124

2. Waypoint Tracking

The waypoint tracking takes place through the motion planner which uses the inverse kinematics. It takes as inputs initial and target position of the robot, the time step and after implementing a feedback structure, computes the right and left velocity and the distance to the target position.
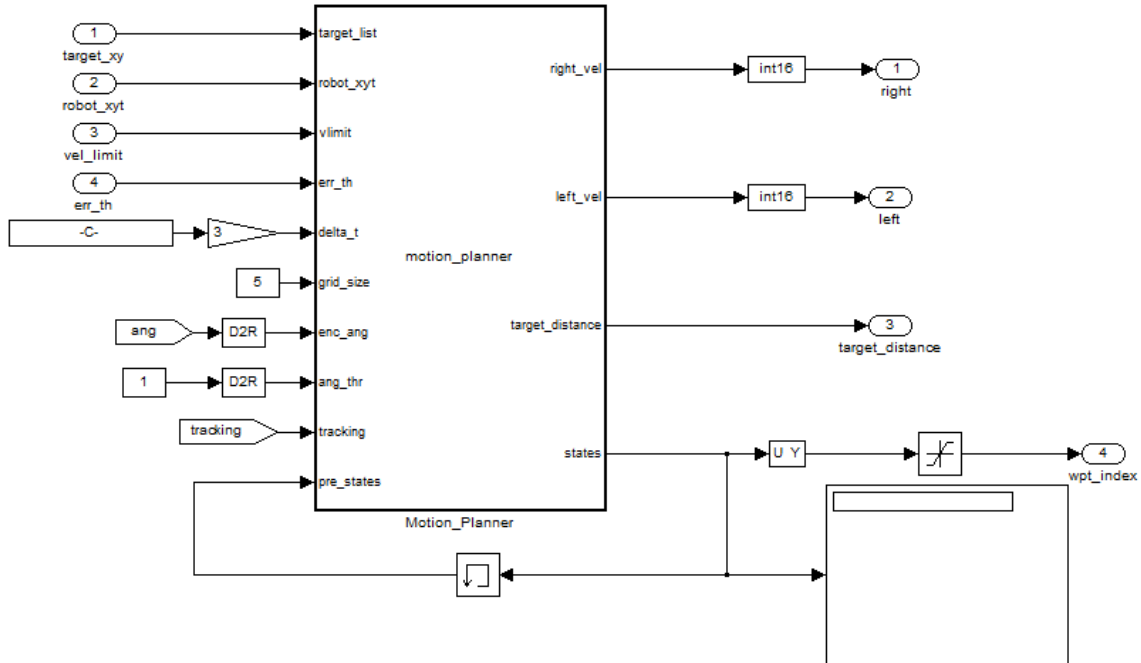
Figure 77.   Motion Planner Simulink Representation

3. Trajectory representation

The trajectory for the waypoint tracking is shown in Figure 79. The Qbot starts from zero point and follows the assigned waypoints. The curves shown in each waypoint is the rotation of the Qbot to the required heading and they are being shown because the diagram tracks the velocity, too. The Qbot computes the right and left wheel velocities required for following the trajectory from one waypoint to another through the inverse kinematics, and with the forward kinematics moves toward each waypoint. It also takes input from the compass and provides feedback. This is the alter to the initial trajectory between waypoints 3 and 4.
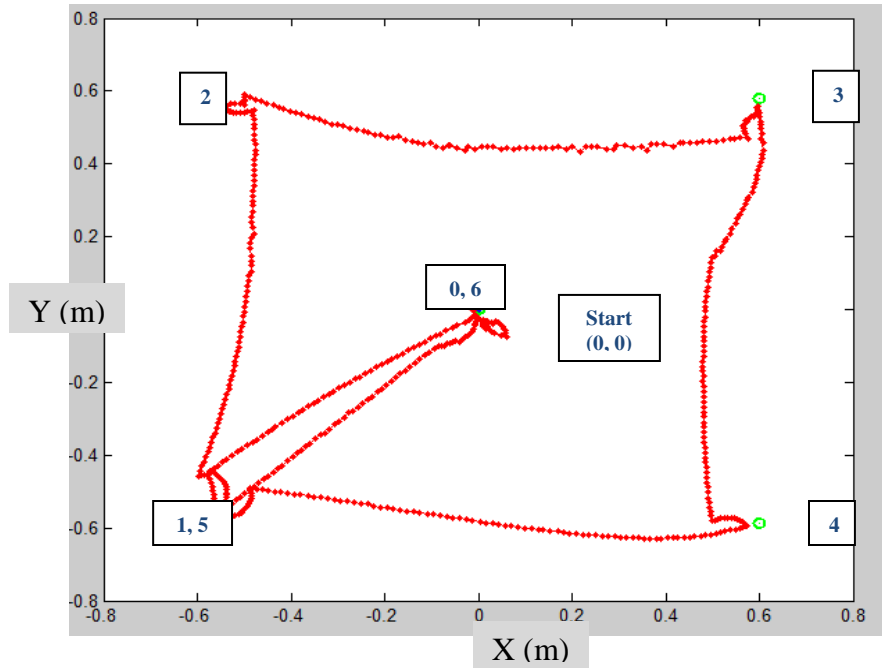
Figure 78.   Qbot Actual Trajectory

4.   Plots

The following plots depict the actual x, y position components and the orientation theta of the Qbot as well as the optitrack measurements of these components for the waypoint pattern described earlier. The difference between the encoded data and the optitrack data is having to do with the fact that the optitrack system captures the motion of the reflector placed on the top of the camera and not the robot itself. The "topt' diagram is so "messy" because the robot adjusts its position and orientation several times till it finds itself within the limits of each waypoint.
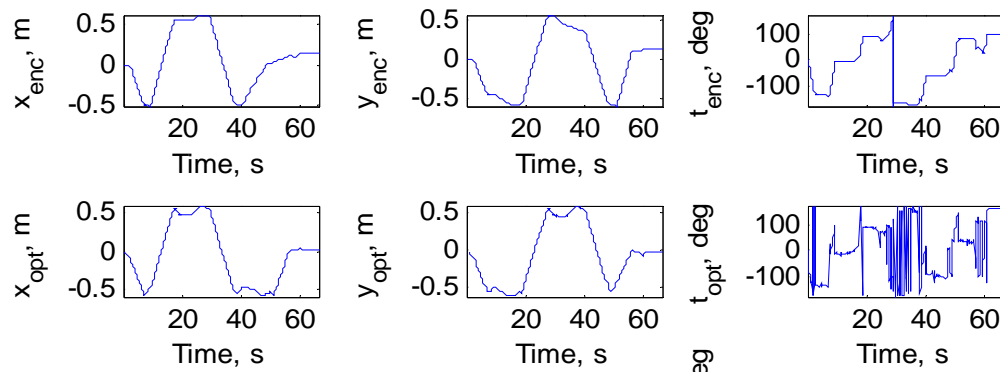
Figure 79.   Qbot x, y, theta plots

126

In order to find the final orientation theta that the robot will apply to go to each waypoint, these components have to be measured. "Dist" is the wheel's distance travelled in a sample time, "Ang" is the angle turned in a sample of time, "heading_abs" and "θmag" are the absolute and magnetic heading measured from the compass model and finally the Distance_t is the distance to the target measured.
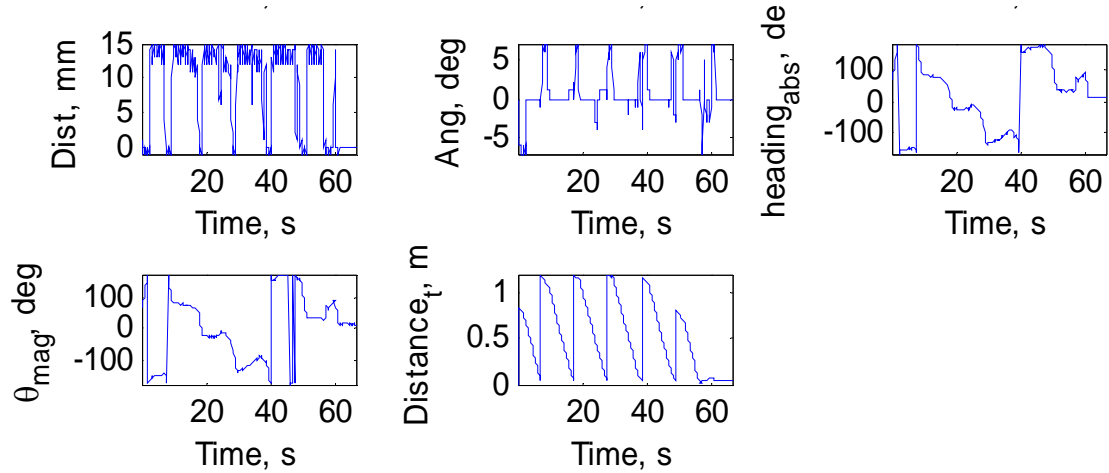
Figure 80.   Qbot heading and distances plots

C.   QBot – Qball COOPERATION WAYPOINT NAVIGATION

The cooperation of the two vehicles will take place in the form of Leader(Qbot) – Follower (QBall). The Qbot navigation will continue taking place as before with the motion planner. In this model though, a real time Data Streamer has to be set in order to send the actual position to the QBall and then the quadrotor starts his trajectory following this positions.
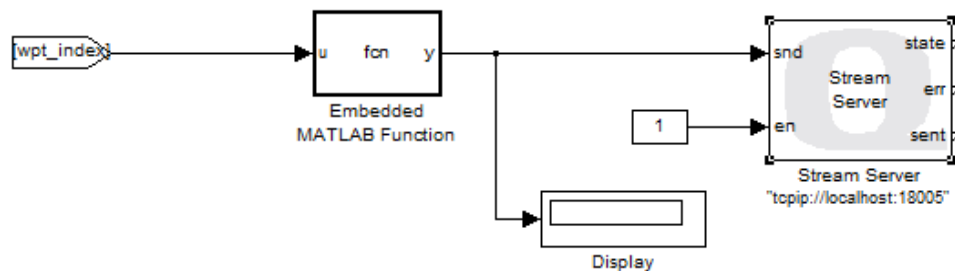
Figure 81.   "Stream to Qball" subsystem

With the above streamer, the waypoint state machine will have as input the coordinates of every waypoint instead of having defined by the user.
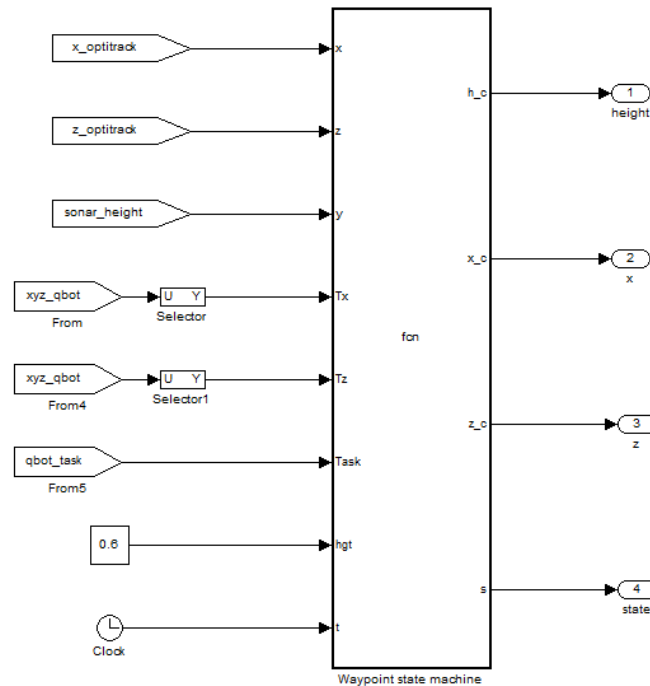
Figure 82.   Qball Waypoint State Machine

The challenge is for the host localization model to recognize both models at the same time and send the actual data to the two controllers. That's why a modification in the Qball and Qbot host localization files needed to be made for the cooperation. The localization models for each vehicle and the modification for their cooperation are shown below:
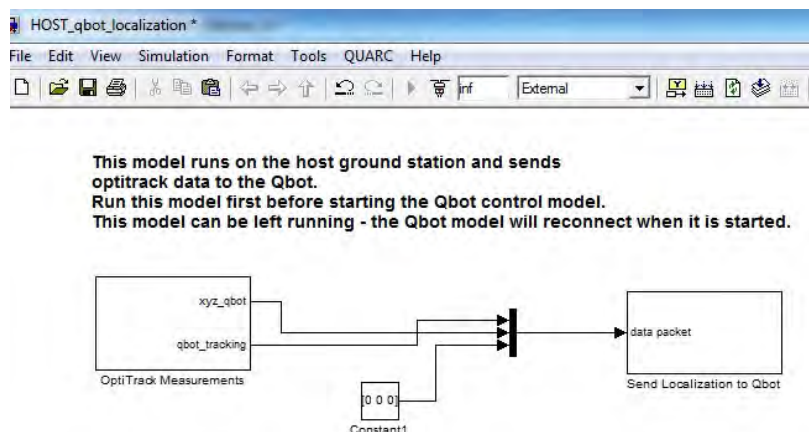
Figure 83.   Host Qbot Localization
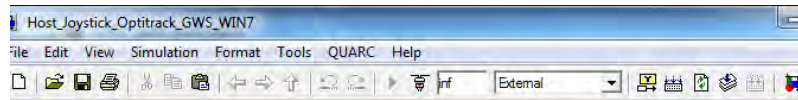
128
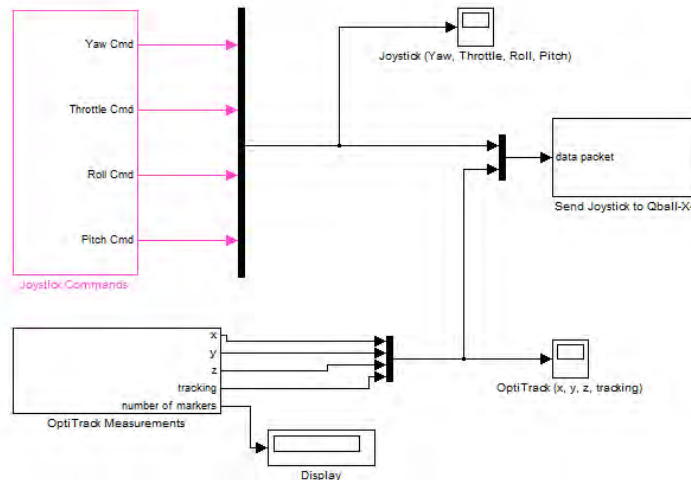
Figure 84.　　Host Qball Localization
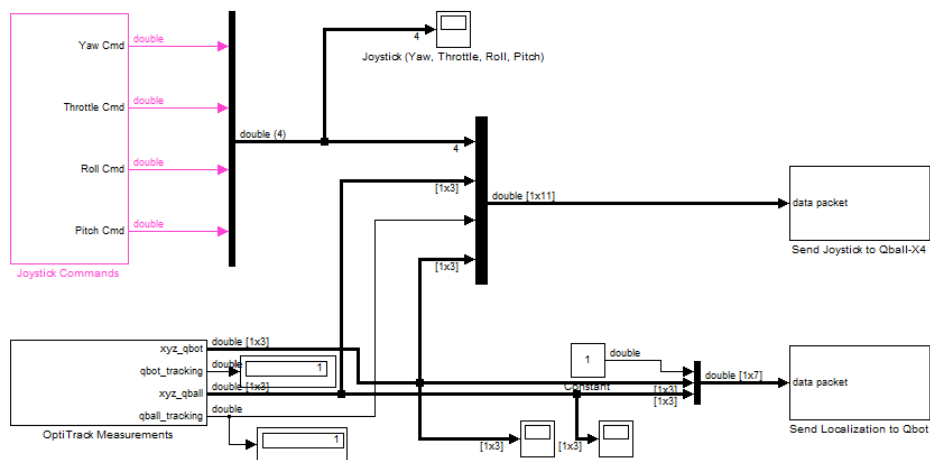
Figure 85.　　Host Qbot - Qball Localization (Modification)

The "Optitrack Measurements"subsystem describes how the localization through the optitrack motion cameras is done for the two models. The calibration file from the cameras is saved in the "Optitrack Point Cloud" Quarc block.

Figure 86.　"Optitrack Measurements" Subsystem

The both vehicles experience the same response as operating alone since the cooperation takes place with the same controllers and waypoint state machines/ motion planners with the difference that the waypoint positions are sent from the Qbot and not form the operator.

VI. CONCLUSIONS – FUTURE WORK

A. CONTRIBUTIONS

This thesis focused on the design of a framework for achieving cooperation between a quadrotor unmanned vehicle, controlled by a LQR controller, and a ground robot applying inverse kinematics as a control strategy. Since not much work has been done for a controller using inverse dynamics for quadrotors UAVs, a direct method for this controller was also introduced.

The main contributions of this thesis are presented as follows:

- The successful set up of the laboratory environment is achieved. This consisted of the localization system with ten motion capture cameras, the ground station, four ground robots and four quadrotors. For this thesis, only one quadrotor and one ground robot was used.

- Applying Newton's laws, the six degree of freedom mathematical model of a quadrotor is derived, including dynamic analysis and design of the control inputs.

- A two dimensional mathematical model for a ground robot is also derived, from the basic physics and robot kinematics.

- A LQR controller is designed for the position and attitude control of the quadrotor, taking place in different decoupled channels and under various flight tests for improvement of their performance.

- Applying forward and inverse kinematics, the control of the ground robot is achieved. The successful data receiving from robot's various sensors is also tested (Camera, IR detectors, Bumper, Sonar detectors).

- A waypoint scenario for the navigation of the two vehicles individually or simultaneously is implemented.

- A proposed controller utilizing direct method with inverse dynamics for real-time control of the quadrotor UAV is partly designed focusing on optimization over quasi optimal solution.

  The ground robot's camera is placed and tested onboard the quadotor as an additional sensor, obtaining successful image.

B.   CONCLUSIONS

The following conclusions can be drawn based on the research of this thesis:

- Quadrotor UAV can be linearized and controlled through linear control methods quite successfully if the flight tests are not far from the hover conditions;

- The LQR controller works well enough when the waypoints distances are not very close to each other (experiences higher overshoot) and not very far away (the trajectory following becomes less precise).

- The LQR controller can be used as a path following controller together with the quasi-optimal trajectory generator for real-time operations achieving better performance toward any disturbances or for obstacle avoidance.

- The quadrotor QBall experiences limitations of operation because of the battery's low capacity. Furthermore, the flight performance is degraded as the battery capacity and life reduces.

- The ground robot's very low speed is a disadvantage for the leader – follower scenario of the two vehicles, forcing the operator to reduce the quadrotor's velocity in order to assist for the accomplishment of the mission.

- The robot's Logitech camera has a slow rate of response, even for the slow velocities of the robot. It is not suggested to be used for faster vehicles like quadrotors unless it is supposed to operate on a standoff basis

C.   FUTURE WORK

The recommendations for future work are:

- Test to the whole length the direct method model and the trajectory generator with LQR controller for different scenarios.

- Achieve the cooperation of two or more quadrotors or two or more ground robots involving collision avoidance scenarios.

- Use multiple quadrotors to observe an object from different angles and all of them arrive simultaneously to allow for military reconnaissance mission, a search and rescue operation, or an industrial inspection routine.

- Implement different controllers for path following instead of LQR. Integral backstepping, a combination of PID and backstepping was proved the best solution for controlling a OS4 quadrotor in EFL's research so it is a possible solution.

- Implement a controller for the non-linear dynamic model of the quadrotor, to obtain the best performance.

- Develop a system for the ground vehicle that it will use a terrain map provided by the quadrotor. The quadrotor can use a added camera or different sensors for image processing in order to construct a map of the terrain around the ground robot so as to navigate to a goal position.

- Finally, since the ground robot consists of a lot of free positions for other sensors, test which other sensors widely used in experiments are compatible with quanser computer, gumstix. This will allow to use these sensors in the quadrotor as well.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX   MATLAB / SIMULINK DOCUMENTATION

setup_qball_x4.M File

A MATLAB script that is run whenever the Qball controller model is opened. This script runs several other scripts to initialize model and controller parameters.

```
% This file runs the calibration and initialization files for the
Qball.
% This file is run automatically when the model is loaded or can be run
% manually.

% Complimentary filter design for roll/pitch measurements.
filter_design;

% LQR gains and other controller parameters for the Qball.
controller_design;
```

filter_design.m File
A script containing the properties of the complementary filter used to estimate the Qball's roll and pitch.

```
t=10;
s = tf('s');
Gg = t^2*s/(t*s+1)^2
Gi = (2*t*s+1)/(t*s+1)^2
```

controller_design.m File
A script used to compute the LQR controller gains used in stabilizing the Qball's orientation and position. This file is run by the setup_qball_x4.m script.

LQR Pitch and Roll Controllers M-file

```
wnom = 15;
L = 0.2;
w = wnom;
K = 120;
J = 0.03;
Jyaw = 0.04;
CLimit = 0.025;
M = 1.4;
g = 9.8;

Am = [0 1 0
      0 0 2*K*L/J
      0 0 -w];
Bm = [0 0 w]';
```

```matlab
Aobs = Am' ;
Bobs = eye(3);
Qobs = diag([.001 10000 .01]);

Robs = diag([ 1 1 1 ])*1;
Kobs = lqr(Aobs,Bobs,Qobs,Robs)
Kobs = Kobs';
Aobs = Aobs'-Kobs*Bobs';
eig(Aobs)
Bobs = [Bm Kobs]
Cobs = eye(3)
Dobs = [ 0 0 0 0
    0 0 0 0
    0 0 0 0];
% augment with integrator
Ai = [Am [0 0 0 ]'
     1 0 0 0 ];
Bi = [Bm' 0]';
Ci = eye(4);
Di = [0 0 0 0 ]';
Q = diag([100 0 22000 10]);
R = 30000;
ki = lqr(Ai,Bi,Q,R);
rp_eig = eig(Ai-Bi*ki);
fprintf ('********************************************* \n');
fprintf('ROLL, PITCH DESIGN \n');
fprintf( 'P = %5.3f D = %5.3f Actuator = %5.3f I = %5.3f \n\n',ki(1),
ki(2),ki(3),ki(4));
for i = 1:4
fprintf('  %5.3f + %5.3f i \n  ',real(rp_eig(i)), imag(rp_eig(i)));
end;
```

LQR Height Controller M-file

```matlab
% Z axis

vlimith = 0.1;
Amh = [0 1
    0 0 ]
Bmh = [0 4*K/M]';
Cmh = [1 0];
Dmh = 0;

% augment with integrator
Aih = [Amh [0  0 ]'
    1  0 0 ];
Bih = [Bmh' 0]';

Cih = eye(3);
Dih = [0 0 0]';

Q = diag([1 0 50]);
R = 5000000;
kh = lqr(Aih,Bih,Q,R);
```

```
h_eig = eig(Aih-Bih*kh);
fprintf ('*********************************************** \n');
fprintf('Z DESIGN \n');
fprintf( 'P = %5.3f D = %5.3f I = %5.3f \n\n',kh(1), kh(2),kh(3));
for i = 1:3
fprintf('   %5.3f + %5.3f i \n   ',real(h_eig(i)), imag(h_eig(i)));
end;
Kph = kh(1);
Kdh = kh(2);
Kwh = 0;
Kih = kh(3);
```

LQR Position Controller M-file

```
tlimit = 5*pi/180; %max pitch cmd radians
%tlimit = 15*pi/180; %max pitch cmd radians
vlimit = 0.3; % max speed cmd in m/sec
%vlimit = 0.5; % max speed cmd in m/sec
Tau_theta = 1/7; % closed loop time constant for pitch response
wt =1/Tau_theta; %closed loop theta bandwidth
kt = 1;
a = [0 1 0 0
     0 0 g 0
     0 0 -wt 0
     1 0 0 0 ];
b = [0 0 wt 0 ]';

q = diag([ 5 2 0 0.1]);
%q = diag([ 5 2 0 0.1]);
%r = 50;
r = 50;

k = lqr(a,b,q,r);

ac = a-b*k;
xy_eig = eig(a-b*k);
Kp = k(1);
Kd = k(2);
Ki = k(4);
Kw = k(3);
fprintf('\n\n X Y Design \n');
fprintf( 'P = %5.3f D = %5.3f Actuator = %5.3f I = %5.3f \n\n',k(1),
k(2),k(3),k(4));
for i = 1:4
fprintf('   %5.3f + %5.3f i \n   ',real(xy_eig(i)), imag(xy_eig(i)));
end;
```

LQR Yaw Controller M-file
```
Ky = 4;
Jy = 0.032;

Amy = [0 1
       0 0 ];
Bmy = [0 4*Ky/Jy]';
```

137

```
Cmy = eye(2);
Dmy = [0;0];

Qy = diag([1 0.1]);
Ry = 1000;
ky = lqr(Amy,Bmy,Qy,Ry);
h_eigy = eig(Amy-Bmy*ky);
Kpyaw = ky(1);
Kdyaw = ky(2);
```

Initialize Qball_Waypoints.m

```
% clear all
close all
clc

height = 301;
width = 301;
axs = [-1 1 -1 1]*0.5;

x = [axs(1),axs(2)];
y = zeros(1, length(x));

figure(1),
plot(x, y),hold on,plot(y, x),
xlabel('X (m)'), ylabel('Z (m)')
% axis([-round(width/2) round(width/2) -round(height/2)
round(height/2)])
axis(axs)
set(gca, 'Ydir', 'reverse')
set(gcf, 'Color', [1 1 1]);

Rx = 0;
Ry = 0;
Rt =0;

n = input('How many waypoints do you want to define? (max 10): ');
if n>10
    n = 10;
end

% X and Y waypoint coordinates
Tx = zeros(11, 1);
Tz = zeros(11, 1);

% Mission state
%    0 : do nothing, wait
%    1 : Goto waypoint
%    2 : Land
Task = ones(11, 1)*2; % Set all tasks to land initially.

for i=1:n
    name = strcat('Enter the coordinate of waypoint no. ', num2str(i));
```

```matlab
        title(name);
        [tx tz] = ginput(1);   % in pix
        plot(tx, tz, 's', 'LineWidth', 2, 'MarkerEdgeColor','k',...
                        'MarkerFaceColor','g',...
                        'MarkerSize',10);
        text(tx+5, tz+15, strcat('Waypoint-', num2str(i)))

        Tx(i) = tx;
        Tz(i) = tz;
        Task(i) = 1;
        if i == n
            Tx(n+1:10) = tx;
            Tz(n+1:10) = tz;
        end
    end
end

hold off
```

QBall Waypoint State Machine function

```matlab
function [h_c, x_c, z_c, s, i] = fcn(x, z, y, Tx, Tz, Task, hgt, t)

% Constants
WP_TOL = 0.1; % Waypoint tolerance, must be less than this distance (m)
to arrive at waypoint.
HGT_TOL = 0.1;
WP_WAIT_TIME = 5; % Wait at waypoint for this many seconds.

% Missions state
%   0 : initialize
%   1 : takeoff
%   2 : goto waypoint
%   3 : land
%   4 : wait at waypoint
persistent state;

persistent wp_index;
persistent to_x;
persistent to_z;
persistent t_start;
persistent t_end;

if isempty(state)
    state = 0;
end
if isempty(wp_index)
    wp_index = 1;
end
if isempty(to_x) || isempty(to_z)
    to_x = 0;
    to_z = 0;
end
if isempty(t_start) || isempty(t_end)
    t_start = 0;
```

```matlab
        t_end = 0;
end

s = state;
i = wp_index;

switch state
    case 0
        h_c = 0;
        x_c = 0;
        z_c = 0;
        if t >= 3
            to_x = x;
            to_z = z;
            state = 1;
        end

    case 1
        h_c = hgt;
        x_c = to_x;
        z_c = to_z;

        if abs(y - hgt) < HGT_TOL
            state = 2;
        end

    case 2
        h_c = hgt;
        x_c = Tx(wp_index);
        z_c = Tz(wp_index);
        if Task(wp_index) == 1
            if sqrt((x - x_c)^2 + (z - z_c)^2) < WP_TOL
                state = 4;
                t_end = t + WP_WAIT_TIME;
                %wp_index = wp_index + 1;
            end
        else
            to_x = Tx(wp_index);
            to_z = Tz(wp_index);
            state = 3;
            t_start = t;
            t_end = t + 3;
        end

    case 3
        if t >= t_end
            h_c = 0;
        else
            h_c = hgt;
        end
        x_c = to_x;
        z_c = to_z;

    case 4
```

140

```
        h_c = hgt;
        x_c = Tx(wp_index);
        z_c = Tz(wp_index);
        if t >= t_end
            state = 2;
            wp_index = wp_index + 1;
        end

    otherwise
        state = 0;
        h_c = 0;
        x_c = 0;
        z_c = 0;
end
```

Initialize Qbot_Waypoints.m

```
close all
clc

load calibration_file

height = 301;
width = 301;
axs = [-1 1 -1 1]*0.6;

x = [axs(1),axs(2)];
y = zeros(1, length(x));

figure(1),
plot(x, y),hold on,plot(y, x),
xlabel('X (m)'), ylabel('Y (m)')
% axis([-round(width/2) round(width/2) -round(height/2)
round(height/2)])
axis(axs)
Rx = 0;
Ry = 0;
Rt =0;

n = input('How many waypoints do you want to define? (max 10): ');
if n>10
    n = 10;
end
Tx = zeros(10, 1);
Ty = zeros(10, 1);
for i=1:n
    name = strcat('Enter the coordinate of waypoint no. ', num2str(i));
    title(name);
    [tx ty] = ginput(1);  % in pix
    plot(tx, ty, 's', 'LineWidth', 2, 'MarkerEdgeColor','k',...
                    'MarkerFaceColor','g',...
                    'MarkerSize',10);
    text(tx+5, ty+15, strcat('Waypoint-', num2str(i)))
    Tx(i) = tx;
```

```matlab
        Ty(i) = ty;
        if i == n
            Tx(n+1:10) = tx;
            Ty(n+1:10) = ty;
        end
    end
end
hold off
```

<u>Qbot Mission Planner function</u>

```matlab
function [right_vel, left_vel, target_distance, states] = ...
    motion_planner(target_list, robot_xyt, vlimit, ...
    err_th, delta_t, grid_size, enc_ang, ang_thr, tracking, pre_states)

% Initialize output variables
right_vel = int16(0);
left_vel = int16(0);
target_distance = -500;
states = pre_states;
if ~(abs(robot_xyt(1)) == 5000 || tracking==0)
% if tracking == 1
    rx = robot_xyt(1);
    ry = robot_xyt(2);
    rtheta = robot_xyt(3);
    target_xy = [rx ry];
    [n xy] = size(target_list);
    if n==2 && xy==1
        target_xy = [target_list(1,1) target_list(2,1)];
        n = 1;
    else
        for i=1:n
            if states(1) == i
                for j=1:xy
                    target_xy(j) = target_list(i, j);
                end
            end
        end
    end
    tx = target_xy(1);
    ty = target_xy(2);

    target_distance = find_dist(rx, ry, tx, ty);
    if ((states(1) == n) && (target_distance <= err_th)) || states(1)
== -1
        if states(5) == 1
            rtheta = 0;
        end
        [ang_to_tar, states(3), states(4)]= find_theta(rx, ry, rtheta,
rx+100, ry, grid_size, ...
            enc_ang, ang_thr, states(3), states(4));
        if states(4) == 1 && states(5) == 0
            [right_vel, left_vel] = solve_inv_kin(0, ...
                ang_to_tar, vlimit, delta_t);
        else
```

142

```matlab
            right_vel = int16(0);
            left_vel = int16(0);
            states(5) = 1;
        end
    else
        if target_distance <= err_th
            states(1) = pre_states(1) + 1;
        else
            [ang_to_tar, states(3), states(4)]= find_theta(rx, ry,
rtheta, tx, ty, grid_size, ...
                    enc_ang, ang_thr, states(3), states(4));
            [right_vel, left_vel] = solve_inv_kin(target_distance, ...
                    ang_to_tar, vlimit, delta_t);
        end
    end
end
return;
%-------------------------------------------------------------------
function [y] = check_angle(x)
y = x;
if x > pi
    y = x - 2*pi;
elseif x < -pi
    y = x + 2*pi;
end
return;
% -------------------------------------------------------------------
function dist = find_dist(rx, ry, tx, ty)
dist = sqrt((rx-tx)^2 + (ry-ty)^2);
return;
% -------------------------------------------------------------------
function [theta, ang_state, rotate_state]= find_theta(rx, ry, rtheta,
tx, ty, ...
    grid_size, enc_ang, ang_thr, pre_ang_state, pre_rotate_state)
X = round((tx - rx)/grid_size);
Y = round((ty - ry)/grid_size);
theta = atan2(Y, X);
if pre_rotate_state == 1
    theta = check_angle(theta - pre_ang_state);
else
    theta = check_angle(theta - check_angle(rtheta));
end
if abs(theta)>=ang_thr
    if pre_rotate_state == 0
        rotate_state = 1;
        ang_state = rtheta;
    else
        rotate_state = pre_rotate_state;
        ang_state = check_angle(pre_ang_state + enc_ang);
    end
else
    rotate_state = 0;
    ang_state = 0;
end
return;
```

```matlab
% ---------------------------------------------------------------------
function [vr, vl] = solve_inv_kin(dist, theta, vlimit, delta_t)
d = 252.5;
vmax = vlimit;%(2);
wmax = (2*vmax)/d;
w = theta/delta_t;
w_sign = sign(w);
if abs(w) > wmax
    w = w_sign*wmax;
    vr = int16(round((d*w)/2));
    vl = int16(-vr);
else
    v = dist/delta_t;
    vr_tmp = (2*v + d*w)/2;
    vl_tmp = 2*v - vr_tmp;

    max_of_vrvl = abs(max(vr_tmp, vl_tmp));
    if max_of_vrvl > vmax
        vr_tmp = (vr_tmp/max_of_vrvl)*vmax;
        vl_tmp = (vl_tmp/max_of_vrvl)*vmax;
    end
    vr = int16(vr_tmp);
    vl = int16(vl_tmp);
end
return;
% ---------------------------------------------------------------------
```

LIST OF REFERENCES

[1]   S. Bouabdallah, "Design and control of quadrotors with application to Autonomous flying," M.S. thesis, Aboubekr Belkaid University, Tlemcen, Algeria, 2007

[2]   http://www.draganfly.com/uav-helicopter/draganflyer-x4/index.php (accessed on 2011/09/11).

[3]   http://vertol.mit.edu/index.html (accessed on 2011/09/11).

[4]   http://www.draganfly.com/news/2007/11/20/vecpav-autonomous-uav-control-systemdraganflyer-helicopters/ (accessed on 2011/09/12).

[5]   D. J. Halaas, S. R. Bieniawski, P. Pigg, and J. Vian, "Control and management of an indoor, health enabled, heterogenous fleet," *Proc. of AIAA Infotech@Aerospace Conference*, Seattle, Washington, 6-9 April, 2009 (AIAA 2009–2036).

[6]   http://users.encs.concordia.ca/~ymzhang/UAVs.htm(accessed on 2011/09/08).

[7]   http://www.draganfly.com/news/2011/06/28/regina-police-department-uses-draganflyer-x6-rc-helicopter-uav-in-homicide-investigation/(accessedon 2011/11/28)

[8]   http://www.draganfly.com/news/2011/02/15/mesa-county-sheriff-colorado-receives-faa-approval-to-operate-the-draganflyer-x6-helicopter-county-wide-2/ (accessed on 2011/11/28)

[9]   G. Hoffmann, H. Haung, S. L. Waslander, and C. L. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Hilton Head, South Carolina, August 2007.

[10]   http://vertol.mit.edu/, (accessed on 2011/09/16).

[11]   http://cmsvt.org/node/19, (accessed on 2011/09/16)

[12]   J. Herkamp. "Deployment of shaped charges by a semi-autonomous ground vehicle." M.S. thesis, Naval Postgraduate School, June 2007.

[13]   P. Castillo, P. Albertos, P. Gracia, and R. Lozano, "Simple real-time attitude stabilization of a quad-rotor aircraft with bounded signals," in *Proceedings of the 45th IEEE Conference on Decision & Control*, San Diego, CA, USA, 2006, pp. 1533 -1538.

[15] A. Soumelidis, P. Gaspar, P. Bauer, B. Lantos, and Z. Prohaszka, "Design of an embedded microcomputer based mini quadrotor UAV," in *Proceedings of the European Control Conference*, Kos, Greece, 2007, pp. 2236–2241.

[16] G. Hoffmann, H. Haung, S. L. Waslander, and C. L. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit,* Hilton Head, South Carolina, August 2007.

[17] I. Sadeghzadeh, A.Mehta, Y.Zhang, and C.-A. Rabbath (2011), "Fault-Tolerant Trajectory Tracking Control of Quadrotor Helicopter Using Gain-Scheduled PID and Model Reference Adaptive Control," presented at the Annual Conference of the Prognostics and Health Management Society 2011 (PHM2011), Sept. 25–29, 2011, Montreal, Quebec, Canada (Nominated as PHM2011 Best Paper).

[18] A. Bani Milhim, Y. M. Zhang, and C.-A. Rabbath (2010), "Gain Scheduling Based PID Controller for Fault Tolerant Control of Quad-Rotor UAV," presented at AIAA Infotech@Aerospace 2010, Atlanta, Georgia, USA, 20-22 April 2010, AIAA Paper 2010-3530.

[19] J. Zhao, W. Sun, Y. Song, and X. Wang, "Fault-tolerant PID controllers design for unknown nonlinear systems based on support vector machine" in *Proceedings of the Control and Decision Conference (CCDC)*, 2010 Chinese, 2010.

[20] A.L. Salih, M. Moghavvemi, H.A.F. Mohamed, K.S. Gaeid, "Modelling and PID controller design for a quadrotor unmanned air vehicle" in, *Proceedings of the 2010 IEEE International Conference on Automation Quality and Testing Robotics (AQTR)*, 2010, Vol.1, pp. 1–5.

[21] E. Altug, J. P. Ostrowski and R. Mahony, "Control of a quadrotor helicopter using visual feedback," *Proceedings of the 2002 IEEE International Conference on Robotics and Automation*, 2002, Vol. 1, pp. 72–77.

[22] D. Lee, T. C. Burg, B. Xian and D. M. Dawson, "Output feedback tracking control of an underactuated quad-rotor UAV," *Proceedings of the 2007 American Control Conference*, New York City, USA, July 11-13, 2007, pp. 1775–1780.

[23] A. Tayebi and S. McGilvary, "Attitude stabilization of a four-rotor aerial robot," in *Proceedings of the 2004 IEEE Conference on Decision and Control*, Atlantis, Paradise Island, Bahamas, 2004, pp. 1216–1221.

[24] A. Benallegue, A. Mokhtari and L. Fridman, "Feedback linearization and high order sliding mode observer for a quadrotor UAV," *Proceedings of the 2006 International Workshop on Variable Structure Systems*, Alghero, Italy, June 5-7, 2006, pp. 365–370.

[25]   I. Kanellakopoulos, P. V. Kokotovic and A. S. Morse, "Systematic design of adaptive controllers for feedback linearizable systems,*" IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, Vol. 36, No. 11, November 1991, pp. 1241–1253

[26]   Khalil, H. K., *Nonlinear Systems*, Upper Saddle River, NJ: Prentice-Hall, Inc., 2nd edition, 1996.

[21]   T. Madani and A. Benallegue, "Backstepping control for a quadrotor helicopter," *Proceedings of the 2006 IEEE/RSJ, International Conference on Intelligent Robots and Systems*, Beijing, China, Oct. 9-15, 2006, pp. 3255–3260.

[28]   T. Madani and A. Benallegue, "Backstepping sliding mode control applied to a miniature quadrotor flying robot," *Proceedings of IEEE Conference on Industrial Electronics*, Paris, France, 2006, pp. 700–705.

[29]   A. A. Mian, M. I. Ahmad and D. B. Wang, "Backstepping based nonlinear flight control strategy for 6 DOF aerial robot," *International Conference on Smart Manufacturing Application*, Kintex, Gyeonggi-do, Korea, April 9-11, 2008, pp. 146–151.

[30]   S. Bouabdallah, P. Murrieri, and R. Siegwart, "Design and control of an indoor micro quad-rotor," *Proceedings of 2004 IEEE International Conference on Robotics and Automation*, New Orleans, USA, 2004, pp. 4393–4398.

[31]   P. Castillo, A. Dzul and R. Lozano "Stabilization of a mini-rotorcraft having four rotors," *Proceedings of the 2004 IEEE International Conference on Industry Technology*, Sendai, Japan, 2004, Vol. 3, pp. 1543–1548.

[32]   K. T. Oner, E. Cetinsoy, M. Unel, M. F. Aksit, I. Kandemir, and K. Gulez, "Dynamic model and control of a new quad-rotor unmanned aerial vehicle with tiltwing mechanism," *Proceedings of World Academy of Science, Engineering and Technology*, November 2008, Vol. 35, pp. 58–63.

[33]   R. Xu and U. Ozguner, "Sliding mode control of a quad-rotor helicopter," *Proceedings of the 45th IEEE Conference on Decision & Control*, Manchester Grand Hyatt Hotel, San Diego, CA, USA, December 13–15, 2006, pp. 4957–4962.

[34]   P. Chandler, "Self-repairing flight control system reliability and maintainability - Executive overview," in *Proceedings of the IEEE National Aerospace and Electronics*, Dayton, OH, 1984, pp. 586–590.

[35]   J. S. Eterno, J. L. Weiss, D. P. Looze, and A. S. Willsky, "Design issues for fault tolerant-restructurable aircraft control," in *Proceedings of the 24th IEEE Conference on Decision and Control*, Ft. Lauderdale, 1985, pp. 900–905.

[36]   Y. M. Zhang and J. Jiang, "Bibliographical review on reconfigurable fault-tolerant control system," *IFAC Annual Review in Control*, 2008, Vol. 32, No. 2, pp. 229–252.

[37]   J. T. Qi, Z. Jiang and X. G. Zhao, "Adaptive UKF and its application in fault tolerant control of quadrotor UAV," *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 20–23, 2007, Hilton Head, South Carolina, pp. 1–15.

[38]   Y. M. Zhang and J. Jiang, "Active Fault tolerant Control System against Partial Actuator Failures," IEEE Proceedings - Control Theory and Applications, January 2002, Vol. 149, No. l, pp. 95–104.

[39]   X. B. Zhang, "Lyapunov-based fault tolerant control of quadrotor unmanned aerial vehicles," M.S. thesis, Concordia University, Montreal, Quebec, Canada, 2010

[40]   A. Bani-Milhim, "Modeling and fault tolerant PID control of a quad-rotor UAV**,"** M.S. thesis, Concordia University, Montreal, Quebec, Canada, 2010

[41]O.A. Yakimenko, "Simplified Modification of the Direct Variational Method for Onboard Solution of Optimization Boundary Problems for Flight Vehicle Trajectories," 1998, *Journal of Computer and Systems Sciences International,* 37(3), pp. 405–415.

[42]   V.N. Dobrokhodov, O.A. Yakimenko, "Synthesis of Trajectorial Control Algorithms at the Stage of Rendezvous of an Airplane with a Maneuvering Object," 1999, *Journal of Computer and Systems Sciences International*, 38(2), pp. 262–277.

[43]   D.V Alekhin, O.A. Yakimenko, "Synthesis of Optimization Algorithm for Route Trajectory by the Direct Variational Method," 1999, *Journal of Computer andSystems Sciences International*, 38(4), pp. 650–666.

[44]   O.A. Yakimenko, "Rapid Onboard Prototyping of Near-Optimal Spatial Trajectories for Pilot's Associate," 2000, Proceedings of the 22nd International Congress of Aeronautical Sciences, Harrogate, United Kingdom, August 27 – September 1.

[45]   O.A. Yakimenko, "Direct Method for Rapid Prototyping of Near-Optimal Aircraft Trajectories," 2000, *AIAA Journal of Guidance, Control, and Dynamics*, 23(5), pp. 865–875.

[46]   I.I. Kaminer, O.A. Yakimenko, A.M. Pascoal, "Coordinated Control of Multiple UAVs for Time-critical Applications," 2006, *Proceedings of the 27th IEEE Aerospace Conference*, Big Sky, Montana, March 4- 11.

[47]   L.-C. Lai et al,"Time-Optimal Control of a Hovering Quad-Rotor Helicopter," *Journal of Intelligent & Robotic Systems*, 2006, Vol. 45, No. 2, pp. 115–135.

[48]   I. D. Cowling, O. A. Yakimenko, and J. F. Whidborne. "A Prototype of an Autonomous Controller for a Quadrotor UAV," in *European Control Conference*, 2007.

[49]   Y. Bouktir, M. Haddad, T. Chettibi, "Trajectory planning for a quadrotor helicopter," in *2008 16th Mediterranean Conference on Control & Automation*, 25–27 June 2008, pp. 1258–1263.

[50]   G. Hoffmann, H. Haung, S. L. Waslander, and C. L. Tomlin, "Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment," in Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit, Hilton Head, South Carolina, August 2007.

[51]   QUANSER INNOVATE EDUCATE, "Quanser Qball-x4 user manual, Document Number 829.

[52]   www.quanser.com, accessed in September, 2011.

[53]   www.Gumstix.com, accessed in September, 2011.

[54]QUANSER INNOVATE EDUCATE, "Quanser Qbot user manual, Document Number 830.

[55]   www.optitrack.com, accessed in September, 2011.

[56]   QUANSER INNOVATE EDUCATE, "Qbot Experiment #01: Kinematics Modeling, Document Number 831.

[57]   D. Halidey, R. Resnik, J. Walker, *Fundamentals of Physics*, John Willey & Sons, Inc, 2008

[58]   G. Dudek, M. Jenkin, *Computational Principles of Mobile Robotics*, Cambridge: Cambridge University Press, 2000.

[59]   R. Murray, Z. Li, and S. Sastry. *A Mathematical Introduction to RoboticManipulation*. Florida: CRC Press, 1994.

[60]   N. Sarkar, V. Kumar, and X. Yun and. Control of mechanical systems withrolling contacts: Applications to mobile robots. *International Journal ofRobotics Research,* 13(1): 55–69, February 1994.

[61]   Benson H. Tongue and Sheri D. Sheppard. *Dynamics: Analysis and Designof Systems in Motion*. John Wiley and Sons, 2005.

[62]   M. J. Nieuwstadt and R. M. Murray, "Approximate trajectory generation for differentially flat systems with zero dynamics," in *Proc. 34th IEEE Conf. Decision Contr.*, New Orleans, LA, December 13–15, 1995.

[63]   V. T. Taranenko, *Experience on Application of Ritz's, Poincare's, andLyapunov's Methods in Solving of Flight Dynamics Problems*. Moscow: Air Force Engineering Academy Press, 1968 (in Russian).

[64]   A. I. Neljubov, *Mathematical Methods of Calculation of Combat, Takeoff/Climb, and Landing Approach Manoeuvres for the Aircraft with 2-D Thrust Vectoring. Flight Characteristics and Combat Maneuvering of Manned Vehicles*. Moscow: Air Force Engineering Academy Press, 1968 (in Russian).

[65]   F. Fahroo and I. M. Ross, "Direct Trajectory Optimization by a Chebyshev Pseudospectral Method," *Journal of Guidance, Control, and Dynamics*, vol.25, no.1, 2002, pp. 160–166.

[66]   J. Vlassenbroeck, and R. Van Dooren, "A Chebyshev Technique for Solving Nonlinear Optimal Control Problems," *IEEE Trans. Autom. Control*, vol.33, no.4, 1988, pp. 333–340.

[67]   M. Huzmezan, G. A. Dumont, W. A. Gough and S. Kovac, "Multivariable Laguerre-based indirect adaptive predictive control a reliable practical solution for process control," in *IASTED Modeling and Control Conf.*, Innsbruck, Austria, February 18–21, 2001.

[68]   M. Fleiss, J. Levine, Ph. Martin and P. Rouchon, "Sur les systèmes non linéarités différentiellement plats," in *C.R. Acad. Sci., Paris*, vol.315, série I, 1992, pp. 619–624.

[69]   A. Chelouah, "Extensions of differential flat fields and Liouvillian systems,"in Proc. 36th IEEE Conf. Decision Contr., San Diego, CA, December 10–12,1997.

[70]   P. Castillo, A. Dzul and R. Lozano, "Real-time stabilization and tracking of a four-rotor mini rotorcraft," *IEEE Transactions on Control Systems Technology*, vol.12, no.4, 2004, pp. 510–516.

[71]   O. Yakimenko. Notes for NPS ME4901(Direct Methods for Rapid Prototyping of Optimal Maneuvers), Naval Postgraduate School, 2001, unpublished.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California